



PUBLISHED FOR SISSA BY SPRINGER

RECEIVED: April 11, 2016

ACCEPTED: May 18, 2016

PUBLISHED: June 9, 2016

Tensor integrand reduction via Laurent expansion

Valentin Hirschi^a and Tiziano Peraro^b

^a*SLAC, National Accelerator Laboratory,
2575 Sand Hill Road, Menlo Park, CA 94025-7090, U.S.A.*

^b*Higgs Centre for Theoretical Physics, School of Physics and Astronomy,
The University of Edinburgh,
Edinburgh EH9 3JZ, Scotland, U.K.*

E-mail: vahirsch@slac.stanford.edu, tiziano.peraro@ed.ac.uk

ABSTRACT: We introduce a new method for the application of one-loop integrand reduction via the Laurent expansion algorithm, as implemented in the public C++ library NINJA. We show how the coefficients of the Laurent expansion can be computed by suitable contractions of the loop numerator tensor with cut-dependent projectors, making it possible to interface NINJA to any one-loop matrix element generator that can provide the components of this tensor. We implemented this technique in the NINJA library and interfaced it to MADLOOP, which is part of the public MADGRAPH5_AMC@NLO framework. We performed a detailed performance study, comparing against other public reduction tools, namely CUTTOOLS, SAMURAI, IREGI, PJFRY++ and GOLEM95. We find that NINJA outperforms traditional integrand reduction in both speed and numerical stability, the latter being on par with that of the tensor integral reduction tool GOLEM95 which is however more limited and slower than NINJA. We considered many benchmark multi-scale processes of increasing complexity, involving QCD and electro-weak corrections as well as effective non-renormalizable couplings, showing that NINJA's performance scales well with both the rank and multiplicity of the considered process.

KEYWORDS: Scattering Amplitudes, Perturbative QCD

ARXIV EPRINT: [1604.01363](https://arxiv.org/abs/1604.01363)

Contents

1	Introduction	1
2	Tensor integrands	3
3	Semi-numerical integrand reduction via Laurent expansion	5
4	Symmetric tensors	8
4.1	Notation	8
4.2	Recursive formulas	9
5	Tensor projectors for Laurent-expansion terms	10
5.1	Numerator evaluation	11
5.2	The μ^2 -expansion	11
5.3	The t_3 -expansion	12
5.4	The t_2 -expansion	13
6	Implementation	13
7	Applications	15
7.1	Timing profile	17
7.2	Stability study	19
8	Conclusions	21
A	Explicit formulas for expansion terms	23
B	Details of timing performances	24
C	Details of stability performances	27
D	Numerical results for benchmark processes	32
D.1	Process $gg \rightarrow t\bar{t}ggg$	33
D.2	Processes $gg \rightarrow Y + \{2, 3\} \cdot g$	34

1 Introduction

Scattering amplitudes in quantum field theory describe the fundamental interactions between elementary particles and provide a powerful way for inferring theoretical models from high-energy phenomenology and vice-versa. At the scales probed by modern colliders, such as the Large Hadron Collider (LHC) at CERN, scattering amplitudes can be computed in

perturbation theory as a Taylor expansion in the coupling constants. Leading-order results are plagued by very large theoretical uncertainties and as such they are often not reliable enough for direct comparisons with experimental results. Phenomenological studies can significantly benefit from theoretical predictions at next-to-leading order accuracy or beyond, which are however complicated by several factors, a crucial one being represented by quantum corrections to amplitudes computed via loop integrals. The calculation of these integrals can be extremely challenging, especially for processes involving many external legs and physical scales. Such processes can however be of great relevance, both for testing the Standard Model in unexplored regions of phase space and for simulating backgrounds to signals of interesting (new) physics. This makes the calculation of loop amplitudes a very active field of research.

A solution to the problem of computing generic one-loop integrals is offered by *integrand reduction* [1–3].

Integrand reduction methods rewrite one-loop integrands as a linear combination of terms in an integrand basis, each of which has five or less loop propagators and yields either a vanishing integral or a known Master Integral. The numerical evaluation of these master integrals is possible by means of public libraries such as ONELOOP [4], GOLEM-95 [5, 6], LOOPTOOLS [7] and QCDLOOP [8] (the last two use the FF library [9] internally). Because the form of the integrand basis is universal and independent of the process or the number of external legs, the algorithm can be applied to any one-loop scattering amplitude in any Quantum Field Theory. The *coefficients* of this decomposition, also known as *integrand decomposition* or *OPP decomposition* [1], can be efficiently computed by evaluating the integrand on *multiple-cuts*, i.e. values of the loop momentum such that a subset of the internal loop propagator denominators vanish. This algorithm is based on repeated numerical evaluations of the integrands and the solution of the resulting subsystems of equations for the coefficients. The method has been implemented in the public codes CUTTOOLS [10] and SAMURAI [11], and has been used within several automated frameworks [7, 12–19] for producing a wide variety of phenomenological results. MADLOOP [20], part of the MADGRAPH5_AMC@NLO [21] framework (abbreviated MG5AMC henceforth), is an example of such tool. It automatically generates one-loop matrix elements and computes them using both traditional OPP reduction (CUTTOOLS and SAMURAI) and tensor integral reduction [22, 23] (as implemented in the tools GOLEM95 [24], PJFRY++ [25] and IREGI). MADLOOP features an in-house implementation of the OPENLOOPS method [16] using a modified version of the ALOHA [26] module to compute the components of the tensor integrand numerator.

More recently, a new approach to one-loop integrand reduction has been developed, namely the *integrand reduction via Laurent expansion* method [27], which elaborates on techniques first proposed in [28, 29] for analytic calculations. Within this approach, the computation of the coefficients of the Master Integrals is significantly simplified by performing a Laurent expansion of the integrands with respect to the components of the loop momentum which are not constrained by the multiple-cut conditions. Since loop integrands are rational functions of the loop components, within semi-numerical calculations the semi-analytic Laurent expansion can be performed via a simplified polynomial division algorithm

between the expansion of the numerator and the loop denominators. Such a technique has been implemented in the public C++ library NINJA [30], which combined to the one-loop package GoSAM [15, 31, 32] has been used for producing several phenomenological results for complicated processes both within the Standard Model and beyond.

The Laurent expansion reduction algorithm implemented in NINJA needs as input procedures which return the leading terms of the above mentioned parametric Laurent expansions of the numerator of the integrand. Generating such input is straightforward and easy-to-automate for analytic one-loop generators such as GoSAM [15, 31] and FORM-CALC [7], but it has so far prevented other one-loop tools following a more numerical approach from using reduction via Laurent expansions.

However, as already noted in refs [27, 30], the only explicit analytic information needed by NINJA about the integrand is its dependence on the loop momentum (and not, for instance, on the external kinematics or polarization states), which is always known in the case of tensor-based loop generators, regardless of whether the entries of the tensors are generated analytically or numerically.

We present an efficient numerical algorithm for constructing the Laurent expansions needed by NINJA directly from the (numerical) entries of loop tensor numerators. In particular, each term of these expansions can be computed by contracting the tensor numerator with specific cut-dependent tensorial projectors. In a numerical implementation, these projectors can in turn be constructed at run-time by means of simple recursive formulas, from lower to higher ranks. This algorithm has been implemented within the NINJA library and the only inputs it needs for the reduction, besides the definition of the loop propagators, are the numerical components of the tensor numerator. This allowed to interface NINJA to MADLOOP whose ability to compute these components is now sufficient, as demonstrated in this paper.

In section 2, we recall the definition of the tensor integrand and in section 3 we briefly review the integrand reduction technique via Laurent expansion. We fix the notation and introduce the computational techniques for building symmetric tensors in section 4 which we use in section 5 to derive formulas for the projection of the tensor numerator onto the coefficients of the Laurent expansion. Details on the implementation of this projection in NINJA as well as its interface to MADLOOP are provided in section 6. We present a detailed study of the stability and timing performances of the combination of MADLOOP and NINJA in section 7 and we give our conclusions in section 8.

2 Tensor integrands

A generic one-loop amplitude can be written as a sum of n -point integrals of the form

$$\mathcal{A} = \int d^d \bar{q} \frac{\mathcal{N}(\bar{q})}{D_1 \cdots D_n}. \quad (2.1)$$

The numerator \mathcal{N} of the integrand is a *polynomial* in the components of the d -dimensional loop momentum \bar{q} , with $d = 4 - 2\epsilon$, while the denominators D_i correspond to Feynman loop propagators and they have the general quadratic form

$$D_i = (\bar{q} + p_i)^2 - m_i^2, \quad (2.2)$$

where p_i is a linear combination of external momenta and m_i is the mass of the particle propagating in the loop (which can be complex-valued when treating unstable particles in the loop within the *complex mass scheme* [33, 34]). One can split the d -dimensional loop momentum \bar{q} into a four-dimensional part q and a (-2ϵ) -dimensional part $\vec{\mu}$,

$$\bar{q} = q + \vec{\mu}, \quad \bar{q}^2 = q^2 - \mu^2. \quad (2.3)$$

The numerator thus becomes a polynomial in the four-dimensional components of q and μ^2 ,

$$\mathcal{N}(\bar{q}) = \mathcal{N}(q, \mu^2). \quad (2.4)$$

We define a four-dimensional *tensor numerator* as a numerator cast into the form

$$\mathcal{N}(q) \equiv \mathcal{N}(q, \mu^2) \Big|_{\mu^2=0} = \sum_{r=0}^R \tilde{\mathcal{N}}_{\mu_1 \dots \mu_r}^{(r)} q^{\mu_1} \dots q^{\mu_r} \quad (2.5)$$

where the tensor coefficients $\tilde{\mathcal{N}}_{\mu_1 \dots \mu_r}^{(r)}$ are independent of the loop momentum. In principle one could consider the more general definition of a d -dimensional tensor numerator, i.e. a linear combination of terms with the form of the r.h.s. of eq. (2.5) multiplied by powers of μ^2 . Although it is straightforward to generalize the results of this paper to d -dimensional numerators, we will only consider the four-dimensional case since tensor-based one-loop matrix element generators typically compute the contributions arising from μ^2 terms in the numerators (also known as R_2 contributions) separately using special tree-level Feynman rules [35].

In the definition in eq. (2.5), a tensor numerator is therefore defined by a sum of tensors homogeneous in rank r , for $r = 0, \dots, R$. The maximum rank R satisfies $R \leq n$ for renormalizable theories, up to a gauge choice. In this paper we will consider a more general case, i.e. $R \leq n + 1$, allowing for up to one effective non-renormalizable vertex in the loop.

In a naive implementation, a generic tensor-numerator of rank R would be defined by $\sum_{r=0}^R 4^r = (4^{R+1} - 1)/3$ entries, growing exponentially with the rank. However, since the tensor described by eq. (2.5) is completely symmetric by construction, we can cast the numerator in the alternative form

$$\mathcal{N}(q) = \sum_{r=0}^R \sum_{\mu_1 \leq \dots \leq \mu_r} \mathcal{N}_{\mu_1 \dots \mu_r}^{(r)} q^{\mu_1} \dots q^{\mu_r} \quad (2.6)$$

where the total number of μ_i -symmetric coefficients $\mathcal{N}_{\mu_1 \dots \mu_r}^{(r)}$ for $r = 0, \dots, R$ (only defined for $\mu_1 \leq \mu_2 \leq \dots \leq \mu_r$) is now $\sum_{r=0}^R \binom{r+3}{r} = \binom{R+4}{R}$, which only grows polynomially (namely as R^4) with the rank.

In order to clarify the notation, we consider the following example of an arbitrarily chosen d -dimensional numerator function

$$\mathcal{N}(\bar{q}) = 4\bar{q}^4 + 2\bar{q} \cdot p_1 \bar{q} \cdot p_2 \bar{q}^2 + 2p_1 \cdot p_2 \bar{q}^2 + 3\bar{q} \cdot p_1 \bar{q} \cdot p_2 + m_X^4 \quad (2.7)$$

which can be recast into the tensorial structure

$$\begin{aligned}
\mathcal{N}(q, \mu^2) = & \underbrace{\{4 g^{\mu\nu} g^{\rho\sigma} + 2 p_1^\mu p_2^\nu g^{\rho\sigma}\}_{\text{sym}}}_{\mathcal{N}_{\mu\nu\rho\sigma}^{(4)}(\{p_i\})} q^\mu q^\nu q^\rho q^\sigma \\
& + \underbrace{\{(2 p_1 \cdot p_2 - 8 \mu^2) g^{\mu\nu} + (3 - 2 \mu^2) p_1^\mu p_2^\nu\}_{\text{sym}}}_{\mathcal{N}_{\mu\nu}^{(2)}(\{p_i\}, \mu^2)} q^\mu q^\nu \\
& + \underbrace{\{4 \mu^4 - 2 p_1 \cdot p_2 \mu^2 + m_X^4\}}_{\mathcal{N}^{(0)}(\{p_i\}, \mu^2)}
\end{aligned} \tag{2.8}$$

where $\{p_i\}$ denotes the collective kinematic information of the phase-space point considered (external momenta, masses, helicity and color assignation, etc...). The $\{\}_{\text{sym}}$ notation indicates that the corresponding tensor components are symmetrized according to the procedure described by eq. (4.1). The four-dimensional tensor numerator is thus obtained by setting $\mu^2 = 0$ and identifying the Lorentz structures multiplying the loop momentum

$$\begin{aligned}
\mathcal{N}(q) = \mathcal{N}(q^2, \mu^2) \Big|_{\mu^2=0} = & \underbrace{\{4 g^{\mu\nu} g^{\rho\sigma} + 2 p_1^\mu p_2^\nu g^{\rho\sigma}\}_{\text{sym}}}_{\mathcal{N}_{\mu\nu\rho\sigma}^{(4)}(\{p_i\})} q^\mu q^\nu q^\rho q^\sigma \\
& + \underbrace{\{2 p_1 \cdot p_2 g^{\mu\nu} + 3 p_1^\mu p_2^\nu\}_{\text{sym}}}_{\mathcal{N}_{\mu\nu}^{(2)}(\{p_i\})} q^\mu q^\nu + \underbrace{\{m_X^4\}}_{\mathcal{N}^{(0)}(\{p_i\})} .
\end{aligned} \tag{2.9}$$

Several techniques have been proposed for constructing the tensor loop numerator. MAD-LOOP, which features an independent implementation of the OPENLOOPS method [16], progressively builds the loop numerator polynomial by successive calls to building block functions numerically computing the loop four-momentum dependence of each vertex and propagator involved in the loop. Contrary to analytic methods, this approach makes it very difficult to reconstruct the d -dimensional μ^2 dependence of the loop numerator, so that it is important that the loop reduction algorithm works equally well with only the 4-dimensional projection $\mathcal{N}(q)$, in which case the missing rational terms arising from μ^2 will be reconstructed independently. An alternative method proposed in ref. [19], addressing the case where a numerical evaluation of the integrand is available but its full polynomial structure is not known, reconstructs the entries of the tensor by sampling the numerator on several values of the loop momentum.

In this paper, we will consider a one-loop tensor integrand to be defined by the entries of the symmetric tensor numerator in eq. (2.6), as well as the momenta p_i and masses m_i appearing in the loop denominators as in eq. (2.2). Thanks to the new projection techniques introduced in this paper, these are now the only input required by NINJA for performing the corresponding loop reduction.

3 Semi-numerical integrand reduction via Laurent expansion

In this section we briefly review the input needed by the semi-numerical integrand reduction via Laurent expansion algorithm [27] as implemented in the C++ library NINJA [30]. We

make no attempt in giving a comprehensive description of this reduction method, which can instead be found in refs [27, 30]. Indeed, in this paper we are merely interested in illustrating how to provide the required input starting from a tensor integrand defined as in section 2, while the internals of the reduction algorithm are unchanged with respect to what has already been presented in the literature.

Integrand reduction methods compute loop integrals by exploiting the knowledge of the algebraic structure of the respective integrands. In more details, any one-loop integrand in dimensional regularization can be written as a sum of contributions with five or less loop propagators, regardless of the number of external legs or the complexity of the process. The corresponding numerators, also known as *residues*, are polynomials with a universal, process-independent parametric form. The unknown process-dependent coefficients appearing in this parametrization can thus be found via a polynomial fit. After integration, the amplitude is expressed as a linear combination of known Master Integrals, whose coefficients can be identified with a subset of the coefficients appearing in the integrand decomposition.

An efficient way of computing the coefficients of the integrand decomposition is by evaluating the integrand on *multiple cuts*, i.e. on values of the loop momentum such that a subset of loop denominators vanish. Indeed, when evaluating the integrand on the generic multiple cut $D_{i_1} = \dots = D_{i_k} = 0$, the only non-vanishing contributions to the integrand decomposition are those coming from the residues sitting on the cut (i.e. vanishing) denominators, as well as from the higher-point residues having the cut denominators as a subset of their propagators. This suggested the possibility of computing the coefficients of each residue by evaluating the integrand on a subset of the solutions of the multiple-cut equations defined by its loop denominators, subtracting the non-vanishing contributions from higher-point residues and solving a system of equations for the unknown coefficients. This is therefore a top-down approach, where higher-point residues are computed first (starting from 5-point residues) and systematically subtracted from the integrand when evaluating lower-point residues. These are known as *subtractions at the integrand level*. This is the approach followed by the public reduction codes CUTTOOLS [10] and SAMURAI [11].

As we mentioned, the integrand reduction via Laurent expansion method [27] can achieve better stability and performance by exploiting the knowledge of the analytic dependence of the integrand on the loop momentum. More specifically, on top of the numerical evaluation of the loop numerator, the algorithm needs as input a numerical evaluation of the leading terms of the numerator with respect to properly defined Laurent expansions, parametric in the directions of the loop momentum unconstrained by the multiple-cut conditions. From these, the coefficients of the integrand decomposition are computed via a simplified polynomial division algorithm between the expansion of the numerator and the loop denominators and corrected by counter-terms depending on higher-point residues. These are referred to as *subtractions at the coefficient level*, which simplify and replace the ones at the integrand level of the original algorithm.

In the following we describe the four inputs needed by NINJA, assuming the rank R satisfies $R \leq n+1$, while in the next sections we will describe how to automatically generate them at run-time from the coefficients of a tensor numerator. In this section we consider a

generic μ^2 -dependent numerator for the sake of generality, although, as already mentioned, we will later specialize to the case of a four-dimensional tensor numerator defined as in eq. (2.6). Notice however that the μ^2 -dependence arising from the expansion of the loop momentum q must be considered in both cases. The four input functions are

- the *numerator function* used for the cut-constructible part of 4-point residues and optional internal tests

$$\mathcal{N}(q, \mu^2), \quad (3.1)$$

as a function of the loop momentum (notice that this is the same input as in traditional integrand reduction algorithms),

- the μ^2 -expansion used for the rational part of 4-point residues, returning the terms $n_{\mu^2}^{(i)}$ defined by the expansion

$$\mathcal{N}(q, \mu^2) \Big|_{q^\nu \rightarrow t v_0^\nu + v_1^\nu, \mu^2 \rightarrow t^2 v_0^2} \stackrel{t \rightarrow \infty}{=} n_{\mu^2}^{(R)} t^R + n_{\mu^2}^{(R-1)} t^{R-1} + \mathcal{O}(t^{R-2}), \quad (3.2)$$

as a function of the four-vectors v_0^ν and v_1^ν

$$n_{\mu^2}^{(i)} = n_{\mu^2}^{(i)}(v_j), \quad (3.3)$$

- the t_3 -expansion used for 3-point and 1-point residues, returning the terms $n_{t_3}^{(i,j)}$ defined by the expansion

$$\mathcal{N}(q, \mu^2) \Big|_{q^\nu \rightarrow v_0^\nu + t v_3^\nu + \frac{\beta + \mu^2}{t} v_4^\nu} \stackrel{t \rightarrow \infty}{=} \sum_{i=R-4}^R \sum_{j=0}^{\lfloor (R-i)/2 \rfloor} n_{t_3}^{(i,j)} t^i \mu^{2j} + \mathcal{O}(t^{R-5}), \quad (3.4)$$

as a function of the four-vectors $v_0^\nu, v_3^\nu, v_4^\nu$ and the complex number β

$$n_{t_3}^{(i,j)} = n_{t_3}^{(i,j)}(v_0, v_3, v_4, \beta), \quad (3.5)$$

- the t_2 -expansion used for 2-point residues, returning the terms $n_{t_2}^{(i,j,k)}$ defined by the expansion

$$\mathcal{N}(q, \mu^2) \Big|_{q^\nu \rightarrow v_1^\nu + x v_2^\nu + t v_3^\nu + \frac{\beta_0 + \beta_1 x + \beta_2 x^2 + \mu^2}{t} v_4^\nu} \stackrel{t \rightarrow \infty}{=} \sum_{i=R-3}^R \sum_{j=0}^{R-i} \sum_{k=0}^{\lfloor (R-i-j)/2 \rfloor} n_{t_2}^{(i,j,k)} t^i x^j \mu^{2k} + \mathcal{O}(t^{R-4}), \quad (3.6)$$

as a function of the four-vectors $v_1^\nu, v_2^\nu, v_3^\nu, v_4^\nu$ and the complex numbers β_i , with $i = 0, 1, 2$

$$n_{t_2}^{(i,j,k)} = n_{t_2}^{(i,j,k)}(v_1, v_2, v_3, v_4, \beta_i). \quad (3.7)$$

We remind the reader that the vectors v_i^ν defining the expansions are cut-dependent, so that the methods for the corresponding coefficients will be called on all the relevant cuts (and possibly more than once per cut, as needed) within one loop reduction. The terms

above are all those needed for calculations with $R \leq n + 1$. If the rank is lower than $n + 1$, fewer terms are needed, and in numerical implementations one should take care that only a minimal number of terms is computed so as to optimize performances.

Any one-loop generator capable of providing a numerical evaluation for the terms in eq. (3.3), (3.5) and (3.7), on top of the evaluation of the numerator as in eq. (3.1), can use NINJA. We now turn to describing a method for building these expansions from a tensor numerator of the form of (2.6). The algorithm then proceeds in a purely numerical way, using as input only the (numerical) entries of a symmetric tensor numerator. Indeed, as already mentioned, the terms of each expansion (3.2)–(3.6) can be defined as contractions between the tensor numerator and cut-dependent tensors which NINJA can build at runtime by means of recursive algorithms. Since all the methods defined above have been implemented within the NINJA library for generic tensor integrands with $R \leq n + 1$, this allows current one-loop tensor generators to use NINJA for the reduction, simply by providing the coefficients $\mathcal{N}_{\mu_1 \dots \mu_r}^{(r)}$ defining the loop numerator as in eq. (2.6).

4 Symmetric tensors

In this section we introduce some notation on symmetric tensors and recursive formulas useful for efficiently building the cut-dependent tensorial projectors appearing in our results.

4.1 Notation

Consider a set of independent vectors $\{v_1 \dots v_k\}$ and the symmetrized tensor product of r (not necessarily distinct) vectors v_{i_1}, \dots, v_{i_r} , with $i_1, \dots, i_r \in \{1, \dots, k\}$, namely

$$\sum_{\sigma \in S_r} v_{i_1}^{\mu_{\sigma(i_1)}} \dots v_{i_r}^{\mu_{\sigma(i_r)}}.$$

This tensor, being completely symmetrized, only depends on the number of times each v_i enters the product. As noted in ref. [36], one can exploit this and introduce a natural correspondence between symmetric tensors and polynomials. More in detail, we will use the following polynomial notation

$$v_1^{r_1} \dots v_k^{r_k} \equiv \frac{1}{r_1! \dots r_k!} \sum_{\sigma \in S_r} v_{i_1}^{\mu_{\sigma(i_1)}} \dots v_{i_r}^{\mu_{\sigma(i_r)}}, \quad (4.1)$$

where r_i is the multiplicity of occurrence of v_i on the r.h.s. of the equation, with $\sum_i r_i = r$. The conventional prefactor we introduced on the r.h.s. exactly cancels out against the equivalent permutations of the tensor indexes, which turns out to be particularly convenient for our application. This is better clarified with a couple of explicit examples

$$\begin{aligned} v_1^r &= v_1^{\mu_1} \dots v_1^{\mu_r} \\ v_1^{r-1} v_2 &= v_2^{\mu_1} v_1^{\mu_2} \dots v_1^{\mu_r} + v_1^{\mu_1} v_2^{\mu_2} v_1^{\mu_3} \dots v_1^{\mu_r} + \dots + v_1^{\mu_1} \dots v_1^{\mu_{r-1}} v_2^{\mu_r}. \end{aligned} \quad (4.2)$$

The notation is also useful for writing tensor relations in a compact way.

As a shorthand, if T and U are symmetric tensors of identical rank r , we also define the contraction

$$T(U) \equiv \sum_{\mu_1 \leq \dots \leq \mu_r} T^{\mu_1 \dots \mu_r} U_{\mu_1 \dots \mu_r}, \quad (4.3)$$

where the sum over repeated indices is restricted so as to be consistent with the definition in eq. (2.6).

4.2 Recursive formulas

Tensors can be built recursively by multiplying lower rank tensors with vectors. For this purpose, we can define the *tensor product* of a rank- $(r-1)$ tensor T with a vector v as the rank- r tensor

$$T \otimes v \equiv T^{\mu_1 \dots \mu_{r-1}} v^{\mu_r}. \quad (4.4)$$

Notice that, even if T is symmetric, the r.h.s. will in general not be a symmetric tensor. However one can easily work out *recursive formulas* which build symmetric tensors from linear combinations of tensor products.

The easiest recursive formula involves rank- r tensors obtained by multiplying a single vector v with itself r times, namely

$$v^r = v^{r-1} \otimes v. \quad (4.5)$$

The easiest non-trivial case involving two vectors v_1 and v_2 is

$$v_1^{r-1} v_2 = v_1^{r-1} \otimes v_2 + (v_1^{r-2} v_2) \otimes v_1. \quad (4.6)$$

The tensor v_1^{r-1} in the first addend on the r.h.s. can in turn be built beforehand using eq. (4.5), while the tensor $v_1^{r-2} v_2$ appearing in the second addend is instead of the same type of the one on the l.h.s. but with a lower rank. Eq. (4.6) can thus be read as a recursive formula for building symmetric tensors of the form $v_1^{r-1} v_2$, where the recursion goes from lower to higher ranks r , starting from $r = 1$ which trivially reads $v_1^0 v_2 = v_2$. A useful generalisation of eq. (4.6) is

$$v_1^{r-k} v_2^k = \left(v_1^{r-k} \otimes v_2^{k-1} \right) v_2 + \left(v_1^{r-1-k} v_2^k \right) \otimes v_1, \quad (4.7)$$

which should be read as a recursion relation in both r and k . Indeed, the second addend involves a symmetric tensor $v_1^{r-1-k} v_2^k$ which is of the same type as the l.h.s. but with total rank $r-1$. The tensor in the first addend may instead be rewritten as $v_1^{r-k} \otimes v_2^{k-1} = v_1^{r'-k'} \otimes v_2^{k'}$, with $r' = r-1$ and $k' = k-1$, hence as a tensor of the same form as the l.h.s. but with lower values of r and k .

For our purposes we need one more recursive formula involving three vectors v_1 , v_2 and v_3 , which reads

$$v_1^{r-k-1} v_2^k v_3 = \left(v_1^{r-k-1} v_2^k \right) \otimes v_3 + \left(v_1^{r-k-1} v_2^{k-1} v_3 \right) \otimes v_2 + \left(v_1^{r-k-2} v_2^k v_3 \right) \otimes v_1. \quad (4.8)$$

The ingredients for this recursion are, similarly as before, tensors with lower r and k , as well as tensors of the form of eq. (4.7).

It is also worth observing that all these recursive formulas can be seen as special cases of a more general one

$$\begin{aligned}
v_1^{k_1} \cdots v_n^{k_n} &= \left(v_1^{k_1-1} v_2^{k_2} \cdots v_n^{k_n} \right) \otimes v_1 \\
&+ \cdots \\
&+ \left(v_1^{k_1} \cdots v_{i-1}^{k_{i-1}} v_i^{k_i-1} v_{i+1}^{k_{i+1}} \cdots v_n^{k_n} \right) \otimes v_i \\
&+ \cdots \\
&+ \left(v_1^{k_1} \cdots v_{n-1}^{k_{n-1}} v_n^{k_n-1} \right) \otimes v_n.
\end{aligned} \tag{4.9}$$

An important observation for numerical calculations is that these recursive formulas have the nice side effect of automatically embedding a system of abbreviations based on reusing common subexpressions. Indeed, as one can see from the definition in eq. (4.4), each entry in a tensor product of total rank r can be obtained from an entry of rank $r - 1$ by a single multiplication. Because our formulas are recursive on the rank and involve linear combinations of tensor products, they provide a built-in mechanism for reusing subexpressions of lower rank when building tensors of higher rank. Moreover, the possibility of reusing common subexpressions is not limited to contributions defined within the same recursive formula, but it can also be extended to contributions across different equations in a way which fits particularly well with the method we will use for building the Laurent expansions of the integrands. We will see in the next section that the leading term of a Laurent expansion can always be obtained from tensors of the form of eq. (4.5). Next-to-leading terms, when needed, will be constructed using eq. (4.6). As we already observed, the r.h.s. of this equation involves a lower-rank tensor of the same form as the l.h.s. and a tensor of the same form of eq. (4.5). While the former is available simply by implementing the recursion from lower to higher ranks, the latter can instead be reused from the tensors recursively built for the leading term. An analogous strategy is also possible for Laurent expansion terms beyond next-to-leading in the Laurent expansion parameter t , where one can always use tensors built in previous steps of the calculation as input for the recursive formulas (more explicit examples will be given in section 5). This greatly reduces the total number of operations needed for the construction of these tensors, without hard-coding complex analytic formulas and while having a relatively simple bookkeeping and still being completely general with respect to the rank of the tensors appearing in the recursion relations.

In the next section we show how the Laurent-expansion terms needed by NINJA can be generated by contracting tensor numerators with tensors of the same kind as those in eq. (4.5), (4.7) and (4.8).

5 Tensor projectors for Laurent-expansion terms

As we stated above, we can build the terms of the Laurent expansions described in section 3 by contracting the tensor numerator with appropriate cut-dependent tensors, which can be seen as projectors. These can in turn be built recursively using the formulas of section 4.2.

We will illustrate the method by explicitly working out a few cases. A complete list of formulas for all the tensor projectors is given in appendix A.

As stated above, in the following we will consider a four-dimensional tensor numerator defined as in eq. (2.6), which thus only depends on the four-dimensional components q^μ of the loop momentum. However, because NINJA implements a d -dimensional version of the integrand reduction method, the parametrization of q on the d -dimensional cut solutions (and thus its Laurent expansion) will still depend on the extra-dimensional variable μ^2 and we must therefore keep track of this dependence while building the expansions (knowing, of course, that it can only come from the loop momentum, and not from the numerator). This d -dimensional reduction yields, on top of the coefficients of the master integrals, the contributions to the rational part of the amplitude coming from the μ^2 dependence of the loop denominators, also known as R_1 . An alternative approach to the calculation of R_1 (used for instance by CUTTOOLS) is its reconstruction from the coefficients of a purely four-dimensional reduction, also known as *cut-constructible* part. It is worth stressing that, as one can observe from the results collected in appendix A, the calculation of the Laurent expansion terms involving μ^2 , with the approach presented in this paper, can always be recycled from the identical terms needed for the cut-constructible part, except for the box residues where the expression is however very simple. This allows to efficiently provide the algorithm of NINJA with all the terms needed by its d -dimensional integrand reduction, while remaining completely agnostic about the μ^2 -dependence of the loop numerator within the reduction routines.

In the following, n is the number of loop propagators and R is the rank of the tensor numerator. The Laurent-expansion parameter is denoted by t , and it is always convenient to compute the terms from higher to lower powers of t . A first reason for this is that the highest powers of t are always needed, while the lower powers might not be. A second compelling reason is that, as we already mentioned, tensors built for the leading terms in t can be reused as input for building terms with lower powers of t , using the recursive formulas introduced in the previous section.

5.1 Numerator evaluation

The easiest function to provide is the evaluation of the numerator function which, using eq. (2.5), simply amounts to

$$\mathcal{N}(q) = \sum_{r=0}^R \mathcal{N}^{(r)}(q^r), \quad (5.1)$$

where we used the notation introduced in section 4 (in particular eq. (4.1) and (4.3)). Each tensor q^r can in turn be built using the recursive formula of eq. (4.5), which here reads

$$q^r = q^{r-1} \otimes q. \quad (5.2)$$

5.2 The μ^2 -expansion

The μ^2 -expansion, only needed for $R \geq n$, involves a single term for $R = n$ and two terms for $R = n + 1$. Since the vector v_0 comes with a power of t , while v_1 is $\mathcal{O}(t^0)$, it is

straightforward to see that the leading term in t of the expansion defined by eq. (3.2) is

$$n_{\mu^2}^{(R)} = \mathcal{N}^{(R)}(v_0^R). \quad (5.3)$$

As done before, we can build v_0^R recursively by means of eq. (4.5)

$$v_0^r = v_0^{r-1} \otimes v_0. \quad (5.4)$$

For the case where $R = n + 1$, we also need the next-to-leading term in t , which is given by the following formula

$$n_{\mu^2}^{(R-1)} = \mathcal{N}^{(R-1)}(v_0^{R-1}) + \mathcal{N}^{(R)}(v_0^{R-1}v_1), \quad (5.5)$$

where the tensor appearing in the first addend of the r.h.s. was already built for the leading terms using eq. (5.4), while for the second addend we can use the recursive relation in eq. (4.6), which in this case reads

$$v_0^{r-1}v_1 = v_0^{r-1} \otimes v_1 + (v_0^{r-2}v_1) \otimes v_0. \quad (5.6)$$

The first addend of this recursive relation also depends on tensors built using eq. (5.4) for the leading term, while the second depends on a lower-rank tensor which gives the recursion in r .

5.3 The t_3 -expansion

The t_3 -expansion has a more complicated structure due to the presence of three vectors (v_0 , v_3 and v_4) and the free variable μ^2 on top of the expansion variable t . More in detail, the vector v_3 comes with a power of t , the vector v_0 is $\mathcal{O}(t^0)$ and the vector v_3 has a $\mathcal{O}(1/t)$ term multiplied by the constant β and a $\mathcal{O}(\mu^2/t)$ term. Hence, the projector for the leading term is a tensor containing only v_3 , while replacing a v_3 by a v_0 decreases the power in t by one, and replacing a v_3 by a v_4 decreases the power in t by two and also adds a μ^2 term.

Since the leading and next-to-leading terms of the expansion do not involve v_4 (and thus neither μ^2), they have exactly the same structure as those for the μ^2 -expansion. They are

$$n_{t_3}^{(R,0)} = \mathcal{N}^{(R)}(v_3^R). \quad (5.7)$$

and

$$n_{t_3}^{(R-1,0)} = \mathcal{N}^{(R-1)}(v_3^{R-1}) + \mathcal{N}^{(R)}(v_3^{R-1}v_0). \quad (5.8)$$

The next-to-next-to-leading terms in t are two, namely a $\mathcal{O}(t^{R-2}\mu^0)$ term and a $\mathcal{O}(t^{R-2}\mu^2)$ term, and their expression involves all the three vectors v_0 , v_3 and v_4 . They are given by the following formulas

$$n_{t_3}^{(R-2,0)} = \mathcal{N}^{(R-2)}(v_3^{R-2}) + \mathcal{N}^{(R-1)}(v_3^{R-2}v_0) + \mathcal{N}^{(R)}(v_3^{R-2}v_0^2) + \beta \mathcal{N}^{(R)}(v_3^{R-1}v_4) \quad (5.9)$$

$$n_{t_3}^{(R-2,1)} = \mathcal{N}^{(R)}(v_3^{R-1}v_4). \quad (5.10)$$

It is worth making a few observations. We already mentioned that the μ^2 -dependent terms can be determined from the cut-constructible ones, and indeed the contribution

$\mathcal{N}^{(R)}(v_3^{R-1}v_4)$ is common between the two equations and thus only needs to be computed once. Moreover the tensor $v_3^{R-1}v_4$, and more in general all those of the form $v_3^{r-1}v_4$, can be computed from the recursion relation in of eq. (4.5) which in this case depends on the tensors v_3^r already computed above for the leading term in t . The tensor $v_3^{R-2}v_0^2$ can instead be computed using the formula in eq. (4.7) with $k = 2$, to be read as a recursion relation in r and depending on tensors of the form $v_3^{r-1}v_0$ already computed for the next-to-leading terms in t .

Similarly, one can compute all the other terms with lower powers of t , by means of a simple power counting on the vectors v_0 , v_3 and v_4 , and building appropriate tensors with the formulas of section 4.2. One can also check that the formulas of eq. (4.5), (4.7) and (4.8) suffice for the calculation of all the terms down to $\mathcal{O}(t^{R-4})$, which is all one needs for integrands with $R \leq n+1$. At each step one can perform the recursion with respect to r , for $r = 0, \dots, R$, reusing as ingredients tensors computed for lower r or for terms with higher powers of t . Explicit formulas for all terms are given in appendix A.

5.4 The t_2 -expansion

The terms for the t_2 -expansion can be computed using the same method as for the t_3 -expansion. Since we are now dealing with three variables (t , x and μ^2) and four independent vectors (v_i , with $i = 1, \dots, 4$) the main difference is a more involved bookkeeping, partially mitigated by the need of less powers in t . Explicit formulas are collected in appendix A.

6 Implementation

The tensor projectors for the Laurent expansion terms described above have been implemented in the NINJA library.

The reduction algorithm implemented in NINJA requires as input a numerator which is an abstract interface implementing the methods described in section 3 (the C++ programming interface is described in ref. [30]). We thus implemented such an interface which computes the expansion terms collected in appendix A from the coefficients $\mathcal{N}_{\mu_1 \dots \mu_r}^{(r)}$ of a generic tensor numerator, defined according to eq. (2.6).

The tensor numerator is treated as a polynomial whose coefficients are stored in a unidimensional array, from lowest to highest according to the *graded lexicographic monomial order* in the variables q^μ with $q^0 \prec q^1 \prec q^2 \prec q^3$ (i.e. terms are ordered by their total degree and terms with the same total degree are ordered lexicographically). This is the same monomial order used internally by MADLOOP and turns out to be particularly convenient for building the tensors described in this paper, since we use formulas which are recursive with respect to the total rank. It is worth observing however that none of the results presented in this paper rely on a specific representation of the momenta (and consequently of the tensor numerator). In particular, the formulas collected in appendix A, as well as the algorithms implemented for building the corresponding tensors, are unchanged after a change of coordinates $q^\mu \rightarrow q'^\mu = \Lambda^\mu_\nu q^\nu$ and can thus be applied to any other representation of the four-dimensional components after converting the momenta v_i used as input into the alternative representation.

We also implemented in NINJA a FORTRAN-90 wrapper exposing this tensor interface, which in principle can be used by any one-loop tensor-based generator by specifying the loop propagators and the coefficients of the tensor numerator defining the integral to be computed. Both the FORTRAN and the C++ interface are publicly available since version 1.1.0 of the library.

MG5AMC (v2.3.4 and onwards) now includes a version of NINJA which is used as the default loop reduction method.¹ This default installation of NINJA can be automatically updated to the latest online one (independently distributed) by running the following command in the MG5AMC interactive interface:

```
MG5_aMC> install ninja
```

NINJA, similarly to CUTTOOLS is available both in double and quadruple precision at runtime and MADLOOP will dynamically switch to quadruple precision when its internal stability tests indicate that the double precision computation does not meet the requirement in numerical accuracy. In MADLOOP, the computation of the tensor numerator coefficients is completely independent from the loop reduction and, as a result, the stability rescue is first attempted by re-performing in quadruple precision *only the loop reduction* (although with input kinematics already promoted to quadruple precision accuracy). This is often enough to restore satisfactory numerical stability, hence avoiding the much more time-consuming full-fledged quadruple precision computation.

We also point out that NINJA's internal kinematic matrix $K_{ij} = (p_i - p_j)^2$, with quantities defined as in eq. (2.2), is initialized directly in MADLOOP where the following three on-shell limits are set to be exact when below a certain adimensional threshold δ set to 10^{-8} by default:

- If $|m_{i(j)}|^2 > 0$, set $K_{ij} = 0$ if $\left| \frac{(p_i - p_j)^2 - m_{i(j)}^2}{m_{i(j)}^2} \right| < \delta$
- If $|E_i + E_j| > 0$, set $K_{ij} = 0$ if $\left| \frac{2(p_i - p_j)}{E_i + E_j} \right| < \delta^2$

This proved to help the numerical stability of the reduction, essentially because it avoids ever approaching the kinematic region where the master integrals switch from a massive to massless description. The choice of the analytic expression to be evaluated by the Master Integral library is typically controlled by an internal infra-red threshold parameter which would apply to each integral independently. By regulating the kinematic matrix in MADLOOP, we guarantee the consistency of the expression employed for all master integrals.

Finally, all loop reduction methods except CUTTOOLS and IREGI can be independently (de-)activated before the generation of the one-loop matrix element numerical code by setting the corresponding MG5AMC path options in `<MG_root>/input/mg5_configuration.txt`. If activated at generation time, then their use at run time can be controlled via the parameter `MLReductionLib` specified in the file `MadLoopParams.dat`.

¹Other alternative reduction algorithms (CUTTOOLS, PJFRY++, GOLEM95, IREGI and SAMURAI) are still used in the event that MADLOOP stability tests deem the kinematic configuration numerically unstable.

Reduction tool	Max. $n_{\text{loop-prop.}}$	Max. rank
NINJA	unlimited	$n_{\text{loop-prop.}} + 1$
SAMURAI	unlimited*	$n_{\text{loop-prop.}} + 1$
CUTTOOLS	unlimited*	$(n_{\text{loop-prop.}} + 1)^\dagger$
IREGI	7	6^{**}
PJFRY	5	$n_{\text{loop-prop.}}$
GOLEM95	6	$\max(6, n_{\text{loop-prop.}} + 1)$

* For reducing loops with 9 (11) loop lines and more, SAMURAI (CUTTOOLS) must be recompiled with an increased value for its default maximum number of denominators.

† Loops with rank $n_{\text{loop-prop.}} + 1$ are supported in CUTTOOLS only for models with effective interactions involving only the Lorentz structures of the Higgs-gluons vertices.

** This IREGI limitation stems from the observation that its reduction of loops with rank larger than 6 is typically unstable for all kinematic configurations.

Table 1. Limitations of the different reduction methods interfaced to MADLOOP. The notation $n_{\text{loop-prop.}}$ refers to the number of internal propagators in the loop considered. All reduction tools except PJFRY support complex masses.

7 Applications

In this section, we will present the summary of a detailed study of the timing and stability performances of MADLOOP interfaced to NINJA. When available, we compare the results obtained with NINJA against other reduction algorithms, namely CUTTOOLS, SAMURAI, IREGI, PJFRY++ and GOLEM95, whose limitations (for the versions interfaced to MADLOOP) are summarized in table 1. CUTTOOLS is a library implementing a four-dimensional version of the integrand reduction method, as well as a reconstruction of the R_1 term as explained at the beginning of section 5. SAMURAI is a similar tool which always performs a full d -dimensional integrand reduction, making it capable of handling d -dimensional loop numerators at the price of being less efficient of four-dimensional ones, since it implements a more complex reconstruction of the integrand. IREGI, PJFRY++ and GOLEM95 are instead tensor integral reduction tools. We stress that MADLOOP can dynamically change at run time the active reduction tool depending on its applicability to each individual (group of) loop(s) being reduced.

The study carried in this section focuses on the following five classes of processes, chosen for their different characteristics that cover a wide spectrum of one-loop matrix-element computations. The notation $\{i, j\} \cdot X$ denotes that we considered all the processes with either i or j occurrences of particle X in the final states.

- A) $gg \rightarrow t\bar{t} + \{0, 1, 2, 3\} \cdot g$

This class of processes is a common benchmark for pure QCD computations as it introduces the top mass as an additional scale. The one-loop amplitudes for each multiplicity of this class of processes were first computed in refs [37–39]. The one-loop matrix element for the process $gg \rightarrow t\bar{t}ggg$ is generated and computed here for the first time for specific kinematic configurations (see appendix. D.1).

- B) $gg \rightarrow H + \{1, 2, 3\} \cdot g$

These processes are computed within the Higgs Effective Interaction Theory as implemented in [40]. In this effective theory the top-quark loop is integrated out, yielding effective interactions between gluons and the Higgs. The resulting dimension-5 operators lead to loops with rank $n_{\text{loop-prop.}} + 1$ which are especially challenging to reduce. Thanks to the trivial Lorentz structure of the effective Higgs interactions, both CUTTOOLS and older versions of SAMURAI are applicable [27], even though they do not support completely general tensor numerators of higher rank. The one-loop amplitudes for each multiplicity of this class of processes were first computed in ref.s [41–43].

- C) $gg \rightarrow Y + \{1, 2, 3\} \cdot g$

This set of processes is similar to the one above, but involving a spin-2 particle Y of mass 1 TeV and whose graviton-like effective interactions are described in section 2.3 of ref. [40] (we considered $\kappa_g = \kappa_q$). In this case, the tensor numerator of the resulting loops with rank $n_{\text{loop-prop.}} + 1$ can have an arbitrary structure that CUTTOOLS cannot handle. The one-loop amplitudes for this class of processes were first computed in [44] and [45] for 0 and 1 additional gluon in the final states and are computed here for the first time for 2 and 3 additional gluons (see appendix. D.2). The study of the phenomenology of QCD corrections within this effective theory featuring a spin-2 particle is in preparation [46].

- D) $gg \rightarrow \{2, 3, 4, 5\} \cdot Z$

This is a class of loop-induced processes for which event generation has recently been automated in MG5AMC [47]. Loop-induced processes are processes without any contribution from tree-level Feynman diagrams, in which case the one-loop amplitude must be squared against itself. This implies that when using integrand reduction (and only then), loops must be reduced individually and independently for each helicity configuration.

Also, given the absence of any Born contribution, loop-induced processes are finite and build by themselves the complete Leading-Order (LO) prediction. For this reason, the speed of event generation and phase-space integration is entirely driven by the one of the one-loop matrix element, making optimizations especially relevant in this case. The gluon-fusion amplitude for $gg \rightarrow ZZ$ was first computed in [48] and results for the processes $gg \rightarrow Z\gamma\gamma$ and $gg \rightarrow ZZZ$ were shown in [47, 49], while the loop-induced processes with four and five final state Z-bosons have never been studied.

- E) $u\bar{u} \rightarrow ZZZZZ, u\bar{u} \rightarrow e^+\nu_e\mu^-\bar{\nu}_\mu b\bar{b}, u\bar{u} \rightarrow t\bar{t}b\bar{b}d\bar{d}$ and $u\bar{u} \rightarrow t\bar{t}b\bar{b}d\bar{d}g$

This less uniform class of process with $u\bar{u}$ initial states serves different purposes. The first process is intended to be compared with its loop-induced counterpart. The second one includes both EW and QCD loop contributions, of all coupling orders, and it probes the behavior of the loop reduction algorithms in the presence of many scales and with complex masses in the loop propagators. The last two processes test the

reduction for high multiplicity processes featuring loops with a large number of loop propagators (up to nine²) but low rank. These four high multiplicity processes have been selected for their specific characteristics from the standpoint of loop reduction and they have no direct phenomenological relevance except for the second one, so that their computation is not present in the literature.

The b-quarks are considered massive in all SM processes except for $u\bar{u} \rightarrow e^+\nu_e\mu^-\bar{\nu}_\mu b\bar{b}$.

7.1 Timing profile

For all processes listed at the beginning of this section, we measure independently the time spent for the computation of the tensor components³ of the loop numerator (t_{num}) and the reduction of the loops (t_{red}), for one random kinematic and helicity configuration summed over color assignments. We stress that t_{red} includes the time spent in evaluating the master integrals as well as in the computation of the coefficients of the Laurent expansion in the case of NINJA.

In MADLOOP, there is no optimization in the computation of the loop integrand tensor numerator across helicity configurations, so that t_{num} scales linearly with the number of non-vanishing helicity configurations. Conversely, t_{red} remains independent of that number since the summation over helicity configurations can be performed before the loop reduction (except for loop-induced processes when using integrand reduction techniques).

We stress that the timing profile for a single helicity configuration is the relevant figure of merit for applications within MG5AMC which does not explicitly sum over helicity configurations for loop contributions, but instead adopts a Monte-Carlo procedure coupled with an adaptive importance sampling.

We summarize our findings in figure 1 showing results obtained with MADLOOP interfaced to either NINJA, SAMURAI or CUTTOOLS.

The x-axis registers the number of loop groups which combines all loops that can be reduced together. This corresponds to the set of loops sharing the same topology (i.e. ordered list of loop propagators), except for loop-induced processes where each loop must be reduced individually and therefore lies in a loop group of its own. Notice that since loops identical up to couplings (like fermion loops of different flavors) are combined already when generating the loop matrix element code, they only count as one.

The main feature of the upper panel of figure 1 is that within each class of processes, the dependence of the reduction time w.r.t. the number of loop groups is linear, as already noticed in [16]. The offset between each class of process is related to the difference in the rank of the constituting loops. The loop rank is typically larger in processes within models with higher dimensional operators (blue hexagons and green squares) as well as in loop-induced processes which involve fermionic loops only (purple triangles). Conversely,

²Note that contrary to NINJA, for enneagons and above SAMURAI must be recompiled after its hard-coded limit on the number of propagators has been increased.

³This timing profile was performed *prior to* an optimization in the computation of the tensor numerator (filtering out some tensor components which are analytically zero) which brings a minor improvement on t_{num} of order $\mathcal{O}(30\%)$ (process dependent). Note that, this does not affect the key comparison of the reduction time t_{red} between different reduction algorithms.

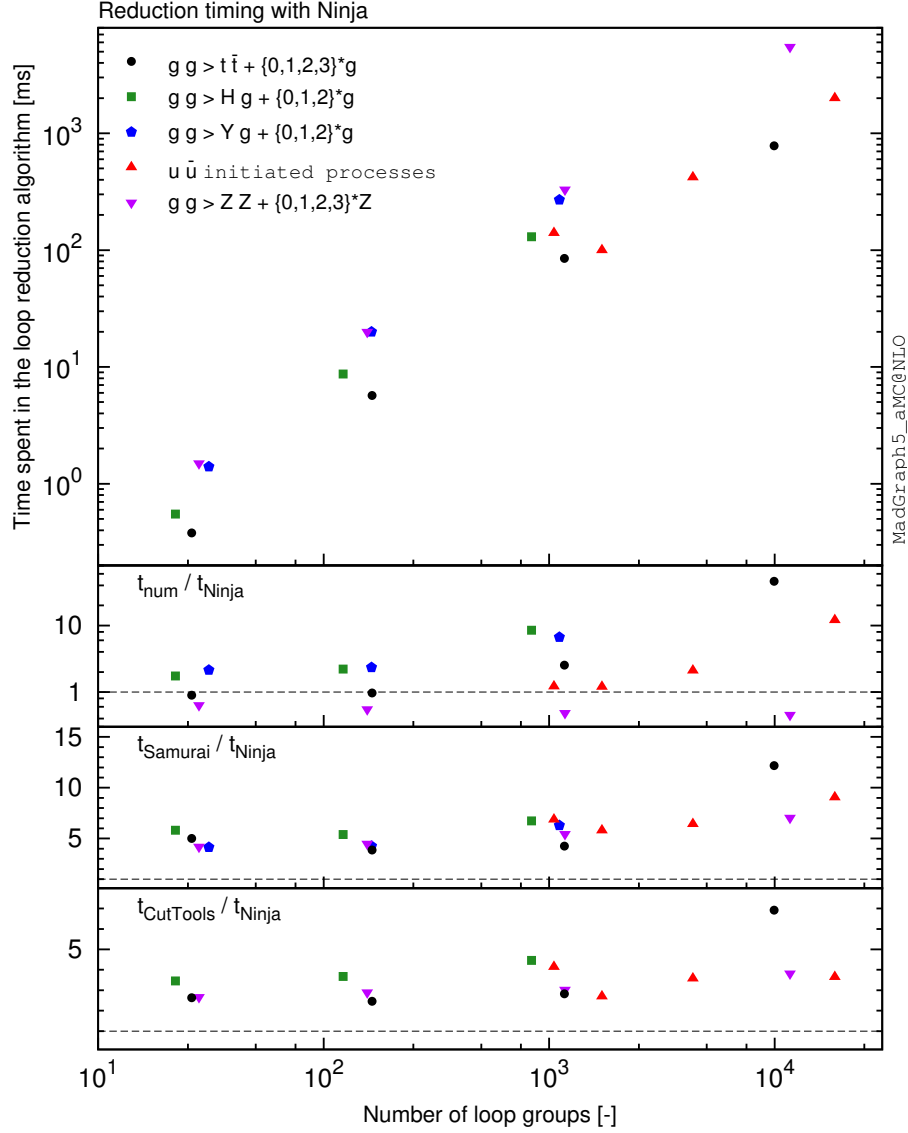


Figure 1. Overview of the timing performances of NINJA, SAMURAI and CUTTOOLS interfaced to MADLOOP on a single core of MacBook (OS 10.8.5), 2.7 GHz Intel Core i7 and using the GNU `gfortran -O2` (v4.8.2) compiler. The timings refer to the computation of the one-loop matrix element summed over color assignments but for a *single helicity* configuration. A loop group combines all loops which can be reduced together. Details on the processes considered are given at the beginning of section 7.

the rank becomes smaller as the number of external fermion lines increases and we indeed observe that the timings for the processes $gg \rightarrow t\bar{t} + n \cdot g$ and $u\bar{u} \rightarrow t\bar{t}b\bar{b}d\bar{d}(g)$ sit on a line underneath (black circles and red triangles). It is interesting to note that the process $u\bar{u} \rightarrow 5 \cdot Z$ is almost two orders of magnitude faster than its loop-induced counterpart, even though they are both contributions to the same final state.

The second inset of figure 1 shows the ratio of the time spent in the computation of the components of the tensor numerator with the loop reduction time with NINJA. This ratio rapidly increases with the multiplicity and number of loop groups, clearly establishing that within the MADLOOP+NINJA implementation, the computation time is asymptotically dominated by the computation of the loop integrand numerator. When no loop grouping is possible, as it is the case for loop-induced processes, we observe the opposite asymptotic behavior hence showing the loop grouping plays an essential role in this limit. We remind the reader that these conclusions apply to the computation of the loop matrix element for a single helicity configuration and only in the context of MADLOOP's technique for the computation of the loop integrand, which is most flexible but less optimal than full-fledged recursion relations [50], especially for processes with large multiplicity.

The bottom two insets of figure 1 compare the performances of the three integrand reduction techniques interfaced to MADLOOP and reveals that NINJA is about 3 to 5 times faster than CUTTOOLS and 5 to 10 times faster than SAMURAI. The reduction time relative to NINJA increases with the process multiplicity, hence assessing the impact of the advantages of the integrand reduction via the Laurent expansion method as the complexity of the considered processes increases.

We include in appendix B a table detailing the timing profile presented in figure 1 as well as process generation times and results obtained with the tensor integral reduction tools IREGI, PJFRY++ and GOLEM95.

7.2 Stability study

We now turn to the assessment of the numerical stability of NINJA for the benchmark processes A)-D) listed at the beginning of this section. We do so by applying the internal stability tests of MADLOOP to a set of N_{PS} random kinematic configurations and we report the resulting accuracy as a cumulative distribution for the fraction of points with a reduction relative accuracy larger than some target Δ on the x-axis.

For the $2 \rightarrow n$ processes, we chose N_{PS} to be 100K for $n=2$, 10K for $n=3$ and 1K for $n > 3$. These N_{PS} kinematic configurations are chosen randomly, with the constraint that all final states satisfy $p_{t,i} > 50$ GeV with angular separation $\Delta R_{ij} = \sqrt{\Delta\phi_{ij}^2 + \Delta\eta_{ij}^2} > 0.5$. The center of mass energy chosen is 1 TeV, except for the processes involving the spin-2 particle Y in which case the center-of-mass energy is set to 1.2 TeV.

MADLOOP combines two stability tests to estimate the numerical accuracy of the result:

- *Loop direction test*: the loop reduction is performed a second time with the order of all propagators reversed (corresponding to the loop momentum redefinition $q \rightarrow -q$) and compared to the original evaluation. This changes the internal numerics of the

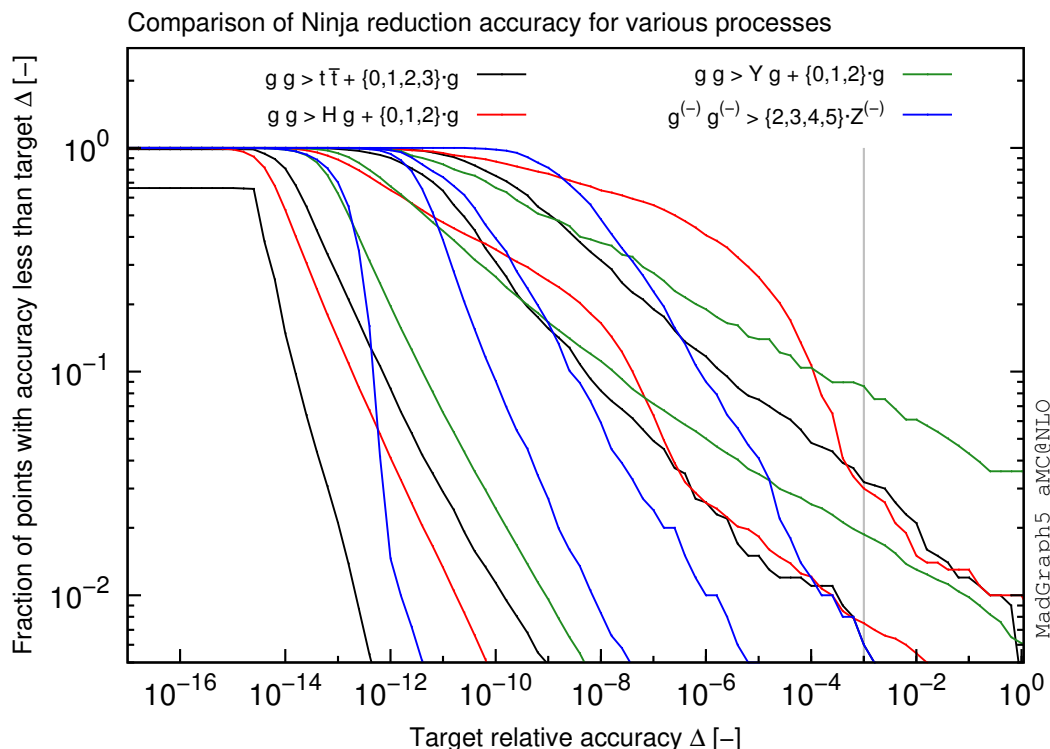


Figure 2. Comparison of the fraction of points with accuracy smaller than the target accuracy Δ on the x-axis obtained with MADLOOP+NINJA (using double precision arithmetics) for a variety of processes (see text for details). The cumulative distributions shown are obtained from kinematic configurations with $\sqrt{s} = 1$ TeV (1.2 TeV for the processes involving the spin-2 particle Y of mass 1 TeV), randomly chosen with the constraints that all final states have a $p_{t,i} > 50$ GeV (except for the loop-induced processes) and an angular separation $\Delta R = \sqrt{\Delta\phi^2 + \Delta\eta^2} > 0.5$. The number of points considered is 100K, 10K, 1K and 1K for processes with 2, 3, 4 and 5 final states respectively. A vertical gray bar is shown at $\Delta = 10^{-3}$ which corresponds to the typical threshold applied during event generation.

reduction, hence assessing its stability. Given that the input kinematics remains unchanged, the tensor numerator components do not have to be recomputed.

- *Lorentz test*: the input kinematic is transformed by a Lorentz rotation for which the loop-matrix element is recomputed and compared to the original one.

Another commonly-used kind of stability test consists in rescaling all dimensionful quantities by a common factor. This is not used by MADLOOP because it proves to be impractical in the general case where the dimension of each of the model input parameters is not necessarily available within the generated code.

The stability tests are performed on a computation of the loop matrix element summed over all helicity configurations, except for the loop-induced processes for which only the *all-minus* helicity configuration is considered.

The vertical gray line at $\Delta = 10^{-3}$ (i.e. 3 stable digits) marks the typical threshold used for Monte-Carlo event generation during which MADLOOP will attempt to rescue the phase-

space points with a numerical stability estimate larger than this target by repeating the reduction (and possibly the computation of the tensor numerator) in quadruple precision. The crossing of the various curves with this gray line therefore gives the fraction of unstable kinematic configurations for which this rescuing procedure will be necessary. For all the processes of highest multiplicity in each class A)-D), this fraction is larger than 1% and almost 10% for $gg \rightarrow Yggg$, which shows that numerical stability becomes an important issue⁴ when attempting the integration of processes with loops of rank 6 and especially 7.

Figure 3 compares the stability of all applicable reduction tools for the processes $gg \rightarrow t\bar{t}gg$, $gg \rightarrow Ygg$ and $u\bar{u} \rightarrow e^+\nu_e\mu^-\bar{\nu}_\mu b\bar{b}$. We observe that NINJA is always the most stable reduction, comparable to that of GOLEM95 which is however considerably slower (see appendix B).

In figure 2, comparing the relative position of the curves for processes with equal number of external legs shows that the determining factor for stability is the tensor numerator rank. This is also manifest when observing that despite the large multiplicity of the process $u\bar{u} \rightarrow e^+\nu_e\mu^-\bar{\nu}_\mu b\bar{b}$ and the complexity of its contributing QCD and EW loops, the stability of its reduction is comparable to that of other processes with a maximum rank of 4.

In an actual Monte-Carlo integration, the phase-space points encountered are not uniformly distributed and as a result the stability profile can potentially be different in this context. For this reason, we also show the stability profiles obtained by considering the kinematics of unweighted events generated at LO accuracy for the LHC14 collider setup using the NNPDF 2.3 (NLO) PDF set [53]. Except for IREGI, we find no qualitative difference and the Monte-Carlo distributions even tend to be slightly more stable for NINJA, showing that its reduction is mainly insensitive to a change of reference frame and c.o.m energy.

In appendix C, we show the stability profiles of processes A)-D) for *all* applicable reduction tools. These further establish the observations drawn in this section.

8 Conclusions

We presented an algorithm for the generation of the expansion terms needed by the *one-loop integrand reduction via Laurent expansion* implemented in the public library NINJA from the numerical components of a tensor numerator. We have shown how, within a numerical calculation, these expansion terms can be obtained by contracting the tensor numerator with appropriate cut-dependent tensors, which in turn can be efficiently built by means of simple recursive relations.

The algorithm has been implemented in the most recent version of the public library NINJA, which can thus be used by tensor-based one-loop generators by providing the numerical entries of the tensor numerator of an integral. We interfaced this library to the MADLOOP generator, part of the MADGRAPH5_AMC@NLO framework (available from v2.3.4 onward).

This allowed us to extensively study the performance and the numerical stability of NINJA and compare it with several other available tools. In terms of reduction speed, we observe that NINJA outperforms all other reduction tools, and in particular SAMURAI and

⁴Faster and more flexible rescue mechanisms may be obtained using expansions around unstable kinematics, as implemented in the private tool COLLIER [51, 52].

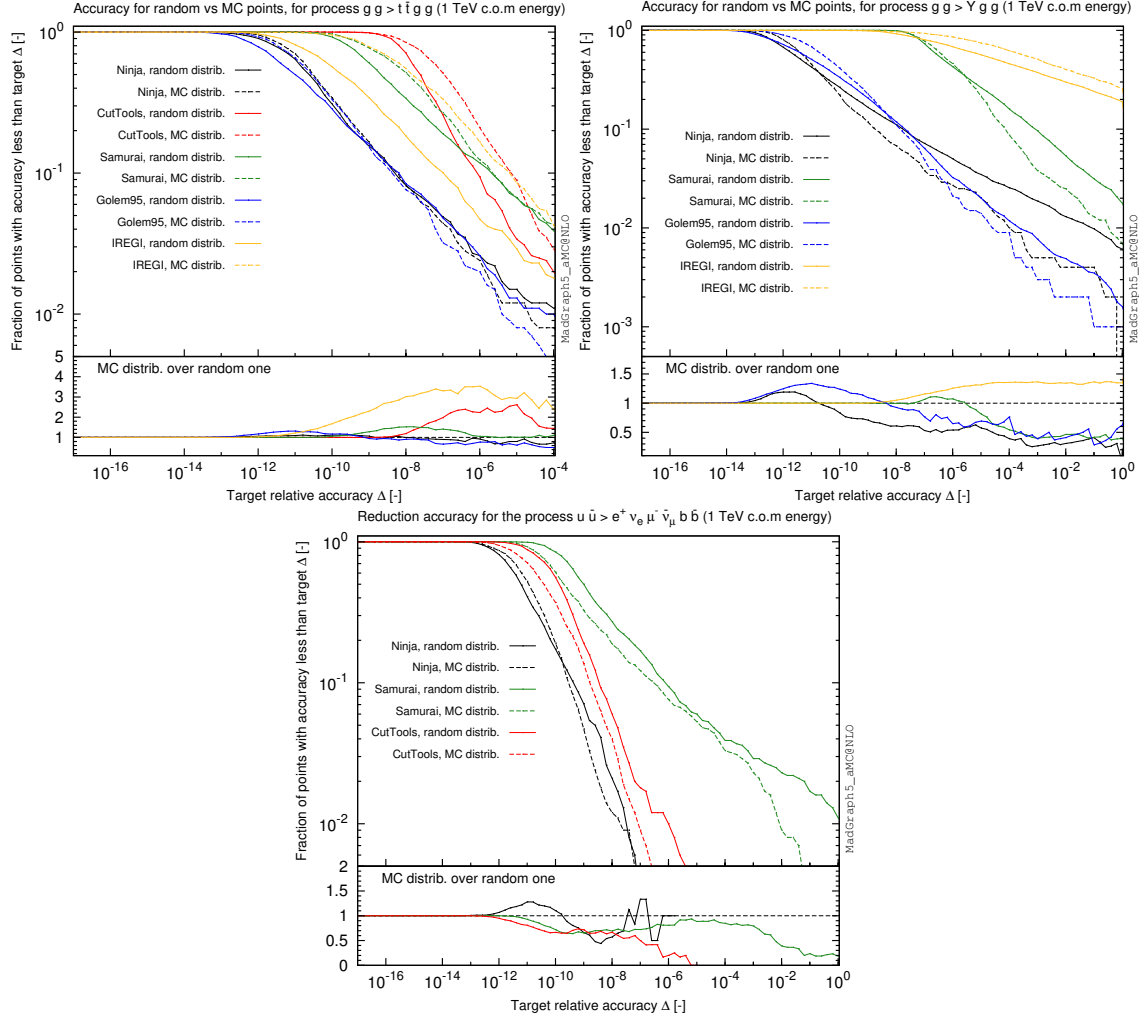


Figure 3. The setup is identical to the one described in the caption of figure 2. The stability profiles obtained from a random distribution of kinematic configurations are compared to the ones obtained from unweighted events generated with LO accuracy at LHC14, using the NNPDF 2.3 (NLO) PDF set.

CUTTOOLS by a factor of about 6 and 3 respectively. Also, NINJA’s improvement over other tools increases with the complexity of the process. In terms of reduction stability, NINJA improves on previous integrand reduction techniques by a considerable amount and in general stands on par with tensor integral reduction as implemented in GOLEM95, which is however more limited and significantly slower. Our results show that numerical instability with NINJA only becomes problematic, because of the slowdown induced by reprocessing unstable points using quadruple precision arithmetics, for loop numerators of rank 7 and above which are not of immediate concern for current phenomenology at particle colliders.

The algorithm and the results presented in this paper therefore enhance the capabilities of MADLOOP and the applicability of NINJA, and will thus be valuable for future high-energy phenomenological studies, especially those involving amplitudes featuring loop diagrams characterized by loop numerators of high rank.

Acknowledgments

We would like to thank Simon Badger, Pierpaolo Mastrolia, Giovanni Ossola, Stefano Frixione and Fabio Maltoni for useful discussions and comments on the draft, and Goutam Das and Huasheng Shao for their feedback on the use of the interface discussed in this paper. The work of TP is supported by an STFC Rutherford Grant ST/M004104/1 and the work of VH is supported by the SNF grant PBELP2 146525.

A Explicit formulas for expansion terms

In this appendix we collect all the formulas for the Laurent expansion terms defined in section 3. The results have been obtained as described in section 5 and implemented in the NINJA library. One can check that all the tensors appearing in the following formulas can be built by means of the recursive relations in eq. (4.5), (4.7) and (4.8). It is worth observing that, for efficiency reasons, in the actual implementation only a minimal set of terms is computed depending on the cut and on the rank of the numerator. We also make sure to reuse contributions to different terms within the same expansion whenever this is possible.

In the following, we again use the notation introduced in section 4.1, with the labeling of the terms in each expansion as defined in section 3. Here is the full list of contributions:

- evaluation of numerator $\mathcal{N}(q)$

$$\mathcal{N}(q) = \sum_{r=0}^R \mathcal{N}^{(r)}(q^r), \quad (\text{A.1})$$

- μ^2 -expansion

$$n_{\mu^2}^{(R)} = \mathcal{N}^{(R)}(v_0^R) \quad (\text{A.2})$$

$$n_{\mu^2}^{(R-1)} = \mathcal{N}^{(R-1)}(v_0^{R-1}) + \mathcal{N}^{(R)}(v_0^{R-1}v_1), \quad (\text{A.3})$$

- t_3 -expansion

$$n_{t_3}^{(R,0)} = \mathcal{N}^{(R)}(v_3^R) \quad (\text{A.4})$$

$$n_{t_3}^{(R-1,0)} = \mathcal{N}^{(R-1)}(v_3^{R-1}) + \mathcal{N}^{(R)}(v_3^{R-1}v_0) \quad (\text{A.5})$$

$$n_{t_3}^{(R-2,0)} = \mathcal{N}^{(R-2)}(v_3^{R-2}) + \mathcal{N}^{(R-1)}(v_3^{R-2}v_0) + \mathcal{N}^{(R)}(v_3^{R-2}v_0^2) + \beta \mathcal{N}^{(R)}(v_3^{R-1}v_4) \quad (\text{A.6})$$

$$n_{t_3}^{(R-2,1)} = \mathcal{N}^{(R)}(v_3^{R-1}v_4) \quad (\text{A.7})$$

$$n_{t_3}^{(R-3,0)} = \mathcal{N}^{(R-3)}(v_3^{R-3}) + \mathcal{N}^{(R-2)}(v_3^{R-3}v_0) + \mathcal{N}^{(R-1)}(v_3^{R-3}v_0^2) + \mathcal{N}^{(R)}(v_3^{R-3}v_0^3) \\ + \beta \left(\mathcal{N}^{(R-1)}(v_3^{R-2}v_4) + \mathcal{N}^{(R)}(v_3^{R-2}v_0v_4) \right) \quad (\text{A.8})$$

$$n_{t_3}^{(R-3,1)} = \mathcal{N}^{(R-1)}(v_3^{R-2}v_4) + \mathcal{N}^{(R)}(v_3^{R-2}v_0v_4) \quad (\text{A.9})$$

$$\begin{aligned}
 n_{t_3}^{(R-4,0)} &= \mathcal{N}^{(R-4)}(v_3^{R-4}) + \mathcal{N}^{(R-3)}(v_3^{R-4}v_0) \\
 &\quad + \mathcal{N}^{(R-2)}(v_3^{R-4}v_0^2) + \mathcal{N}^{(R-1)}(v_3^{R-4}v_0^3) + \mathcal{N}^{(R)}(v_3^{R-4}v_0^4) \\
 &\quad + \beta \left(\mathcal{N}^{(R-2)}(v_3^{R-3}v_4) + \mathcal{N}^{(R-1)}(v_3^{R-3}v_0v_4) + \mathcal{N}^{(R)}(v_3^{R-3}v_0^2v_4) \right) \\
 &\quad + \beta^2 \mathcal{N}^{(R)}(v_3^{R-2}v_4^2)
 \end{aligned} \tag{A.10}$$

$$\begin{aligned}
 n_{t_3}^{(R-4,1)} &= \mathcal{N}^{(R-2)}(v_3^{R-3}v_4) + \mathcal{N}^{(R-1)}(v_3^{R-3}v_0v_4) + \mathcal{N}^{(R)}(v_3^{R-3}v_0^2v_4) \\
 &\quad + 2\beta \mathcal{N}^{(R)}(v_3^{R-2}v_4^2)
 \end{aligned} \tag{A.11}$$

$$n_{t_3}^{(R-4,2)} = \mathcal{N}^{(R)}(v_3^{R-2}v_4^2), \tag{A.12}$$

• t_2 -expansion

$$n_{t_3}^{(R,0,0)} = \mathcal{N}^{(R)}(v_3^R) \tag{A.13}$$

$$n_{t_3}^{(R-1,0,0)} = \mathcal{N}^{(R-1)}(v_3^{R-1}) + \mathcal{N}^{(R)}(v_3^{R-1}v_1) \tag{A.14}$$

$$n_{t_3}^{(R-1,1,0)} = \mathcal{N}^{(R)}(v_3^{R-1}v_2) \tag{A.15}$$

$$\begin{aligned}
 n_{t_3}^{(R-2,0,0)} &= \mathcal{N}^{(R-2)}(v_3^{R-2}) + \mathcal{N}^{(R-1)}(v_3^{R-2}v_1) + \mathcal{N}^{(R)}(v_3^{R-2}v_1^2) \\
 &\quad + \beta_0 \mathcal{N}^{(R)}(v_3^{R-1}v_4)
 \end{aligned} \tag{A.16}$$

$$n_{t_3}^{(R-2,0,1)} = \mathcal{N}^{(R)}(v_3^{R-1}v_4) \tag{A.17}$$

$$n_{t_3}^{(R-2,1,0)} = \mathcal{N}^{(R-1)}(v_3^{R-2}v_2) + \mathcal{N}^{(R)}(v_3^{R-2}v_1v_2) + \beta_1 \mathcal{N}^{(R)}(v_3^{R-1}v_4) \tag{A.18}$$

$$n_{t_3}^{(R-2,2,0)} = \mathcal{N}^{(R)}(v_3^{R-2}v_2^2) + \beta_2 \mathcal{N}^{(R)}(v_3^{R-1}v_4) \tag{A.19}$$

$$\begin{aligned}
 n_{t_3}^{(R-3,0,0)} &= \mathcal{N}^{(R-3)}(v_3^{R-3}) + \mathcal{N}^{(R-2)}(v_3^{R-3}v_1) \\
 &\quad + \mathcal{N}^{(R-1)}(v_3^{R-3}v_1^2) + \mathcal{N}^{(R)}(v_3^{R-3}v_1^3) \\
 &\quad + \beta_0 \left(\mathcal{N}^{(R)}(v_3^{R-2}v_1v_4) + \mathcal{N}^{(R-1)}(v_3^{R-2}v_4) \right)
 \end{aligned} \tag{A.20}$$

$$n_{t_3}^{(R-3,0,1)} = \mathcal{N}^{(R)}(v_3^{R-2}v_1v_4) + \mathcal{N}^{(R-1)}(v_3^{R-2}v_4) \tag{A.21}$$

$$\begin{aligned}
 n_{t_3}^{(R-3,1,0)} &= \mathcal{N}^{(R-2)}(v_3^{R-3}v_2) + \mathcal{N}^{(R-1)}(v_3^{R-3}v_1v_2) + \mathcal{N}^{(R)}(v_3^{R-3}v_1^2v_2) \\
 &\quad + \beta_0 \mathcal{N}^{(R)}(v_3^{R-2}v_2v_4) + \beta_1 \left(\mathcal{N}^{(R)}(v_3^{R-2}v_1v_4) + \mathcal{N}^{(R-1)}(v_3^{R-2}v_4) \right)
 \end{aligned} \tag{A.22}$$

$$n_{t_3}^{(R-3,1,1)} = \mathcal{N}^{(R)}(v_3^{R-2}v_2v_4) \tag{A.23}$$

$$\begin{aligned}
 n_{t_3}^{(R-3,2,0)} &= \mathcal{N}^{(R-1)}(v_3^{R-3}v_2^2) + \mathcal{N}^{(R)}(v_3^{R-3}v_1v_2^2) + \beta_1 \mathcal{N}^{(R)}(v_3^{R-2}v_2v_4) \\
 &\quad + \beta_2 \left(\mathcal{N}^{(R)}(v_3^{R-2}v_1v_4) + \mathcal{N}^{(R-1)}(v_3^{R-2}v_4) \right)
 \end{aligned} \tag{A.24}$$

$$n_{t_3}^{(R-3,3,0)} = \mathcal{N}^{(R)}(v_3^{R-3}v_2^3) + \beta_2 \mathcal{N}^{(R)}(v_3^{R-2}v_2v_4). \tag{A.25}$$

B Details of timing performances

This section presents MADLOOP timing profile for the generation and the computation of the tensor numerator components for all the benchmark processes A)-D) introduced at the beginning of section 7. We also show the time necessary for performing the loop

Timing profile	$u\bar{u} \rightarrow 5 \cdot Z$	$u\bar{u} \rightarrow e^+ \nu_e \mu^- \bar{\nu}_\mu b\bar{b}$	$u\bar{u} \rightarrow t\bar{t} b\bar{b} d\bar{d}$	$u\bar{u} \rightarrow t\bar{t} b\bar{b} d\bar{d} g$
Generation time	45 min	2h10m	16 min	~2 days
# Loop diagrams	3870	51363	5875	99981
# Loop groups	1051	4343	1716	18469
Maximum numerator rank	6	4	4	5
Integrand computation time	0.17 s	0.88 s	0.12 s	24 s
NINJA reduction time	0.14 s	0.42 s	0.10 s	2.0 s
SAMURAI reduction time	0.96 s	2.7 s	0.58 s	18.1 s
CUTTOOLS reduction time	0.58 s	1.5 s	0.27 s	7.3 s
	$gg \rightarrow 2 \cdot Z$	$gg \rightarrow 3 \cdot Z$	$gg \rightarrow 4 \cdot Z$	$gg \rightarrow 5 \cdot Z$
Generation time	15.6 s	110 s	26 min	29 h
# Loop diagrams	28	156	1176	11700
# Loop groups	28	156	1176	11700
Maximum numerator rank	4	5	6	7
Integrand computation time	0.95 ms	11 ms	0.17 s	2.5 s
NINJA reduction time	1.5 ms	20 ms	0.32 s	5.5 s
SAMURAI reduction time	6.3 ms	90 ms	1.8 s	39 s
CUTTOOLS reduction time	4.0 ms	58 ms	1.1 s	21 s
IREGI reduction time	31 ms	1.9 s	120 s	N/A
GOLEM95 reduction time	20 ms	0.94 s	28 s	N/A
PJFRY++ reduction time	5 ms	0.36 s	N/A	N/A

Table 2. The upper table presents results for processes with high-multiplicity and low loop numerator ranks. Notice that the process $u\bar{u} \rightarrow e^+ \nu_e \mu^- \bar{\nu}_\mu b\bar{b}$ (massless b -quarks) includes *all* SM tree and loop contributions (i.e. of both QCD and EW origin, resonant as well as non-resonant ones and also including contributions of order $\mathcal{O}(\alpha_s^0)$). The timing of the loop matrix element $2\Re(\mathcal{A}^{(loop)} \mathcal{A}^{(tree)\dagger})$ of the process $u\bar{u} \rightarrow ZZZZZ$ (denoted $u\bar{u} \rightarrow 5 \cdot Z$ in the table) echoes the profiling presented in the lower table for the evaluation of the loop-induced matrix element $|\mathcal{A}^{(loop)}|^2$ of the gluon fusion contributions up to the same final states. All timings in this table refer to the computation of the loop matrix element summed over colors but for a *single* helicity configuration. The test machine is using a single core (for process generation as well) of an Intel Core i7 CPU (2.7 GHz) and the executable is compiled with GNU `gfortran -O2` (v4.8.2).

reduction with each of the six reduction tools interfaced to MADLOOP (when available): IREGI, PJFRY++, GOLEM95, CUTTOOLS, SAMURAI and NINJA. The reader can easily reproduce analogous results for various compilers and machines by using the automated `check timing` command of the MG5AMC interface.

The numbers shown in tables 2 and 3 refer to the computation of the loop matrix element averaged and summed over color assignments but *for a single helicity* and kinematic configuration. This is the relevant figure of merit in MG5AMC since it implements a Monte-Carlo over helicity (with importance sampling) when integrating loop contributions.

Timing profile	$gg \rightarrow t\bar{t}$	$gg \rightarrow t\bar{t}g$	$gg \rightarrow t\bar{t}gg$	$gg \rightarrow t\bar{t}ggg$
Generation time	14 s	46 s	14 min	~2 days
# Loop diagrams	36	427	5547	82470
# Loop groups	26	164	1168	9940
Maximum numerator rank	3	4	5	6
Integrand computation time	0.34 ms	5.5 ms	215 ms	36 s
NINJA reduction time	0.38 ms	5.7 ms	85 ms	0.78 s
SAMURAI reduction time	1.9 ms	22 ms	362 ms	9.5 s
CUTTOOLS reduction time	1.0 ms	14 ms	240 ms	5.4 s
IREGI reduction time	5.5 ms	290 ms	12.8 s	13 min
GOLEM95 reduction time	4.6 ms	113 ms	3.4 s	N/A
PJFRY++ reduction time	1.1 ms	50 ms	N/A	N/A
	$gg \rightarrow Hg$	$gg \rightarrow Hgg$	$gg \rightarrow Hggg$	
Generation time	28 s	122 s	1h50m	
# Loop diagrams	69	875	12300	
# Loop groups	22	122	836	
Maximum numerator rank	5	6	7	
Integrand computation time	0.96 ms	19.2 ms	1.1 s	
NINJA reduction time	0.55 ms	8.7 ms	130 ms	
SAMURAI reduction time	3.2 ms	47 ms	876 ms	
CUTTOOLS reduction time	1.9 ms	32 ms	580 ms	
IREGI reduction time	18.6 ms	1.7 s	N/A	
GOLEM95 reduction time	6.9 ms	0.28 s	N/A	
	$gg \rightarrow Yg$	$gg \rightarrow Ygg$	$gg \rightarrow Yggg$	
Generation time	72 s	234 s	3h20m	
# Loop diagrams	141	1463	18420	
# Loop groups	31	163	1111	
Maximum numerator rank	5	6	7	
Integrand computation time	3.0 ms	47 ms	1.8 s	
NINJA reduction time	1.4 ms	20 ms	0.27 s	
SAMURAI reduction time	5.8 ms	85 ms	1.7 s	
IREGI reduction time	35.4 ms	3.5 s	N/A	
GOLEM95 reduction time	12.5 ms	0.50 s	N/A	

Table 3. Same setup as described in the caption of table 2. Profiling of the runtime of the processes $gg \rightarrow \{X\} + n \cdot g$ with $\{X\} = t\bar{t}, H, Y$ and $n = (0,)1, 2, 3$. The symbol Y denotes a spin-2 particle with a mass of 1 TeV and interactions as described in section 2.3 of ref. [40].

It is worth noting however that for loop matrix elements with a Born contribution, only the numerator computation time (t_{num}) scales with the number of contributing (analytically non-zero) helicity configurations (n_{hel}) whereas the reduction time (t_{red}) remains constant as the integrand numerators can be summed over helicity configurations before being reduced. The total time for the computation of the loop matrix element summed over all helicity configurations can therefore simply be computed as

$$t_{\text{tot, hel+color summed}} = n_{\text{hel}} \cdot t_{\text{num}} + t_{\text{red}} \quad (\text{B.1})$$

since MADLOOP does not implement optimizations across different helicity configurations.

For loop-induced processes however [47], this is not possible when using reduction at the integrand level, in which case both t_{num} and t_{red} scale with n_{hel} .

The number of loop diagrams indicated does not count the multiple copies with different quark flavors in the loop. A loop group refers to a group of loops which can be reduced together. This number is equal to the number of loops for loop-induced processes since each loop must be reduced individually in this case; otherwise it regroups all loops sharing the same topology (ordered list of denominators identified by their mass m_i and four-momentum flows p_i).

The synthetic figure 1 of the main text illustrates best the results and we find that NINJA outperforms all reduction tools considered for all the benchmark processes. We note however that the advantageous apparent exponential growth of the integrand reduction time with the process multiplicity is mitigated by the factorial growth of the time spent in computing the numerator tensor components. This is intrinsic to MADLOOP's approach based on Feynman diagrams which offers maximal flexibility at the expense of not taking advantage of the optimal scaling behavior of recursion relations [50]. We stress that MADLOOP implements a caching system for recycling part of trees and loops shared across different diagrams. This emulates what recursion relations achieve, but only to a lesser extent even though it already considerably improves the computation time of the tensor numerator.

Generation time is usually not considered as relevant given that it must be performed once per process only. In practice however, this can be of concern since it is typically not easily parallelizable and is also a general hinderance when it comes to testing, debugging or quickly exploring the impact of some modifications to a model. In MADLOOP's approach, generation time is hardly an issue for current phenomenologically relevant processes, but its growth is such that we reached the limit of reasonable process generation for $gg \rightarrow t\bar{t}gg$ which requires about 2 days of sequential runtime and 40 GB of RAM.

C Details of stability performances

In this appendix, we present the stability profiles obtained for processes A)-D) described at the beginning of section 7 and for the six loop reductions tools interfaced to MADLOOP (when available): IREGI, PJFRY++, GOLEM95, CUTTOOLS, SAMURAI and NINJA. The reader can easily reproduce these profiles by using the automated `check stability` command of the MG5AMC interface.

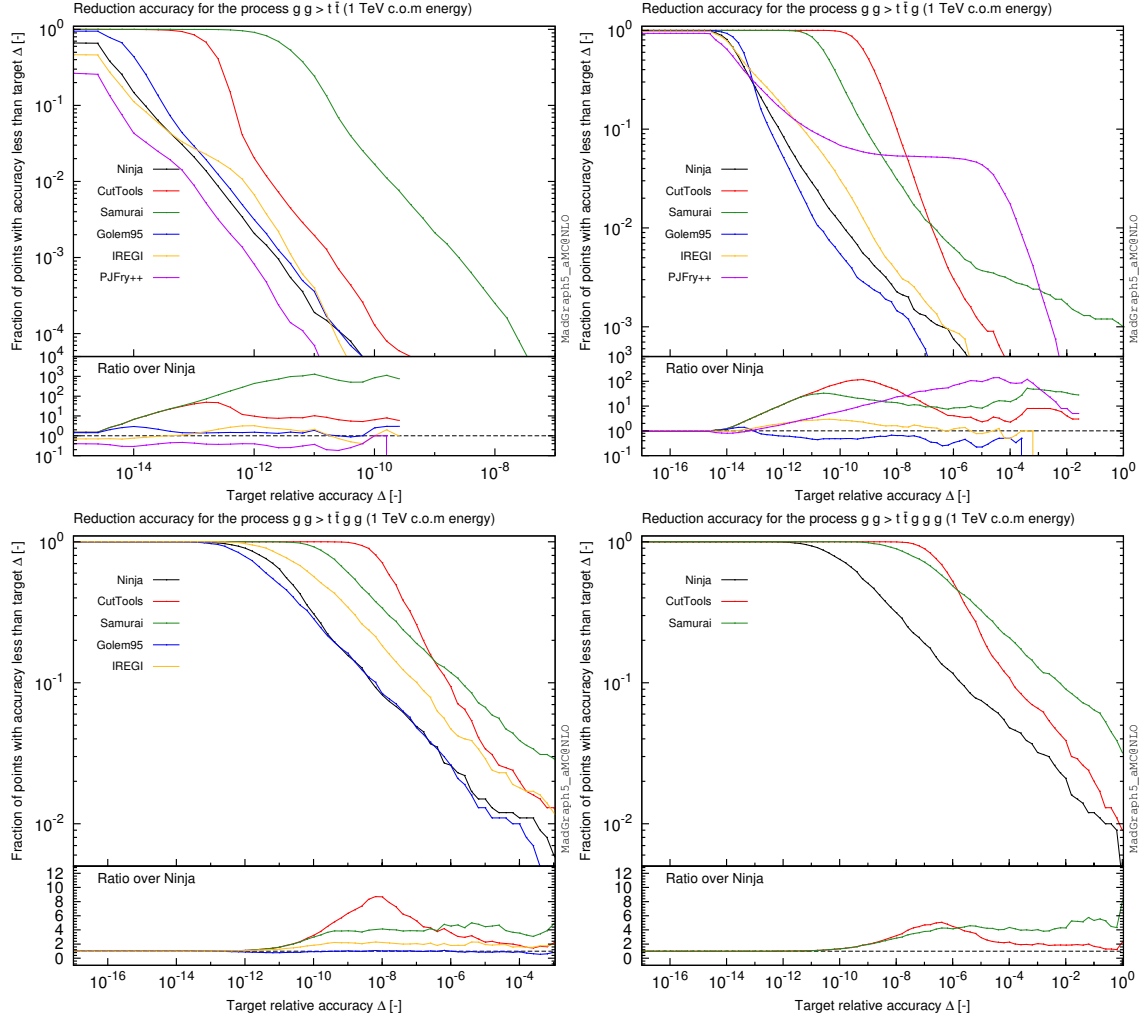


Figure 4. Same setup as in figure 2, showing the stability profile of all applicable reduction methods interfaced to MADLOOP for the class of processes $gg \rightarrow t\bar{t} + \{0, 1, 2, 3\} \cdot g$.

Figure 4 shows results for the process $gg \rightarrow t\bar{t} + \{0, 1, 2, 3\}g$. The plateau displayed by PJFry++ in $gg \rightarrow t\bar{t}g$ is reminiscent of a known issue in this tool which is now no longer maintained.

Figure 5 shows results obtained for the processes $gg \rightarrow H/Y + \{1, 2, 3\} \cdot g$ involving effective interactions yielding loops of numerator ranks equal to one plus the number of loop denominators. Such loops are particularly challenging to reduce and we observe that the accuracy deteriorates significantly when assuming a completely general form of the higher rank tensor, as it is the case in $gg \rightarrow Y + n \cdot g$, compared to the simpler tensor numerators obtained in $gg \rightarrow H + n \cdot g$. Indeed, the residues and consequently the fitting procedure are considerably more involved for higher rank integrands. It is worth noticing that for the simpler tensor structure of $gg \rightarrow H + n \cdot g$ however, the Laurent expansion method is significantly more accurate than other integrand reduction tools, since the numerator expansion methods will return zeroes for vanishing higher rank coefficients, thus avoiding

the (inexact) numerical reconstruction of such zeroes from multiple evaluations of the integrand. In the Higgs case, the stability profile exhibits an unusual shape with a very steep dependence with Δ in the region $[10^{-5}, 10^{-3}]$, indicating that any decrease of the default Monte-Carlo stability threshold of $\Delta = 10^{-3}$ would have a significant impact on runtime performances as the fraction of points that must be reprocessed using quadruple arithmetics increases. The comparison of the two profiles obtained using tensor integral reduction highlights the importance of the internal numerical stabilization mechanisms of Golem95 as the rank of the loop numerator increases.

Figure 6 shows stability results for processes with high multiplicity but relatively low rank. We observe that even though the processes $u\bar{u} \rightarrow t\bar{t}b\bar{b}d\bar{d}(g)$ involve up to 8-(9)-loops, their numerical stability is only slightly worse than that of the lower multiplicity processes with equal maximum rank, such as $gg \rightarrow t\bar{t}g(g)$. It is interesting to note that when the multiplicity is larger than the rank by several units, the numerical stability of CutTools and NINJA is almost identical, as expected from the fact that the integrand reduction via Laurent expansion method and the traditional OPP numerator fitting are very similar in this limit. Indeed, when the multiplicity n of the loop lines and the rank r of the numerator satisfy $r \leq n - 4$, one can easily show that the result is only determined by the cut-constructible coefficients of the boxes. These in turn, given their simplicity, are the only coefficients that NINJA computes with the same algorithm as traditional integrand reduction.

The two processes of the bottom insets of figure 6 introduce new scales in the loop amplitudes; first with external massive lines for $u\bar{u} \rightarrow 5 \cdot Z$ for which CutTools is slightly more stable than NINJA (unlike for all other processes) and secondly with internal (complex) massive lines for the complete QCD+EW loop contributions to the process $u\bar{u} \rightarrow e^+\nu_e\mu^-\bar{\nu}_\mu b\bar{b}$ for which NINJA is more stable. From the numerical stability standpoint, the interesting characteristics of loop-induced processes such as $gg \rightarrow n \cdot Z$ is that they only involve fermion loops of maximal numerator rank. Contrary to the stability studies performed on all other classes of processes, we investigated the loop-induced ones by considering only a single helicity configuration; either $g^{(-)}g^{(-)} \rightarrow n \cdot Z^{(-)}$ (all-minus) or $g^{(-)}g^{(-)} \rightarrow n \cdot Z^{(0)}$ (all longitudinal Z bosons). We find a similar stability behavior for both helicity configurations, except for the lowest multiplicity process.

The peculiarity of the $gg \rightarrow ZZ$ process is that its fermion box loop contribution becomes unstable when the transverse momentum p_t of the final state Z-bosons tends to zero (i.e. all external momenta aligned on the beam axis). Given the constrained $2 \rightarrow 2$ kinematics, this configuration is often probed which is also why this process is typically integrated using a technical (very) small cut in the Z-boson p_t . The upper left plot of figure 7 reveals that in this $p_t \rightarrow 0$ limit, the fermion box stability significantly depends on the helicity configuration considered.⁵ For more than two Z-bosons in the final states, this dependence is much weaker, and mainly reflects the difference in the size of the relative contribution of the more stable Higgs channels.

⁵The numerical stability of the process $gg \rightarrow ZZ$ is actually minimal for the helicity configuration ‘-+00’ (and its parity-related counterpart ‘+-00’), not shown in figure 7.

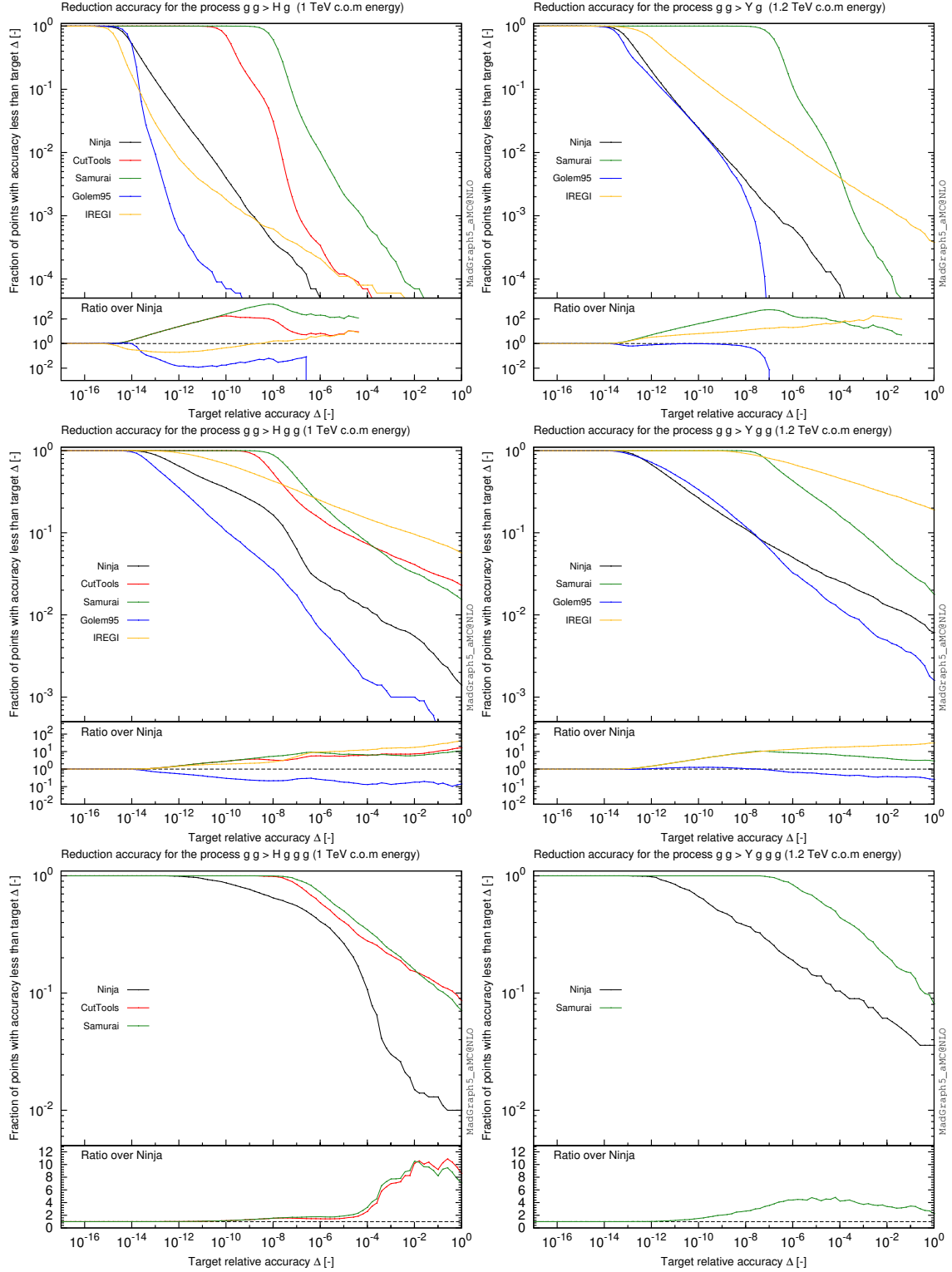


Figure 5. Same setup as in figure 2. The plots on the left hand side study processes involving a Higgs in the Higgs Effective Theory model described in ref. [40]. The plots on the right hand side study processes involving a spin-2 particle denoted Y with a mass of 1 TeV and interactions as described in section 2.3 of ref. [40].

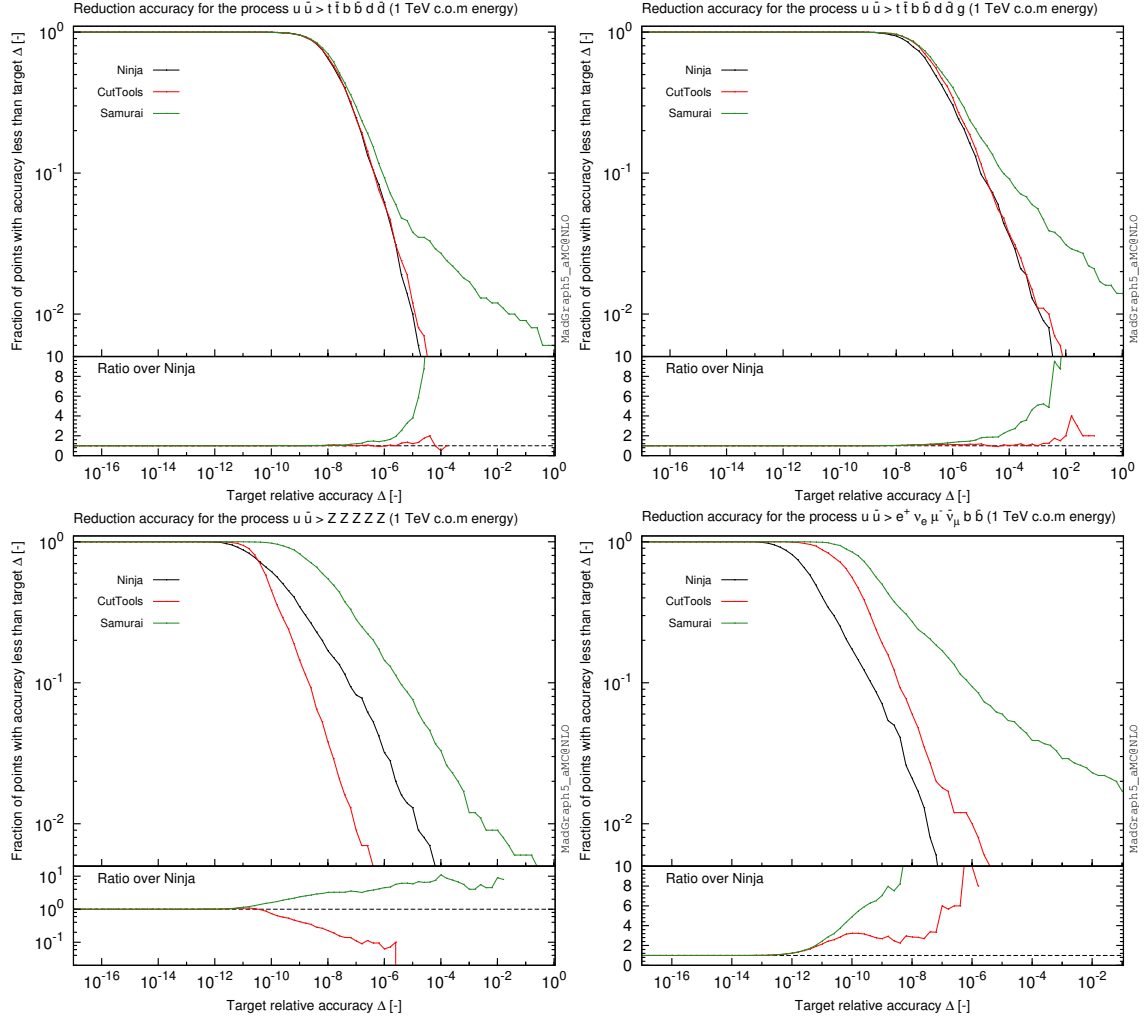


Figure 6. Same setup as in figure 2 but for two processes with 6 or more final states. Notice that the process $u\bar{u} \rightarrow e^+ \nu_e \mu^- \bar{\nu}_\mu b\bar{b}$ with massless b -quarks (right panel) includes *all* SM tree and loop contributions (i.e. of both QCD and EW origin, resonant as well as non-resonant ones and also including contributions of order $\mathcal{O}(\alpha_s^0)$).

Even though the highest multiplicity loop-induced process is of maximal rank 7, it is significantly more stable than $gg \rightarrow Y g g g$ which shares the same maximal rank but with lower multiplicity. On the other hand, its stability is on par with the other $2 \rightarrow 5$ process $gg \rightarrow t\bar{t} g g$ which has many more diagrams but with a maximal rank of only 6.

In summary and in terms of numerical stability, NINJA performs better or at least as well as the best other public reduction tool available for all considered processes except two. First, contrary to Golem95, NINJA has around 1% of unstable kinematic configurations (featuring less than 3 digits) for $gg \rightarrow H g g$ and secondly, it is slightly less stable than CUTTOOLS for $u\bar{u} > 5 \cdot Z$.

The observations drawn in this appendix suggest that the numerical stability of one-loop matrix elements can be *classified* according to mainly their maximal rank and multiplicity. However, a large difference in the number and complexity of the contributing loops

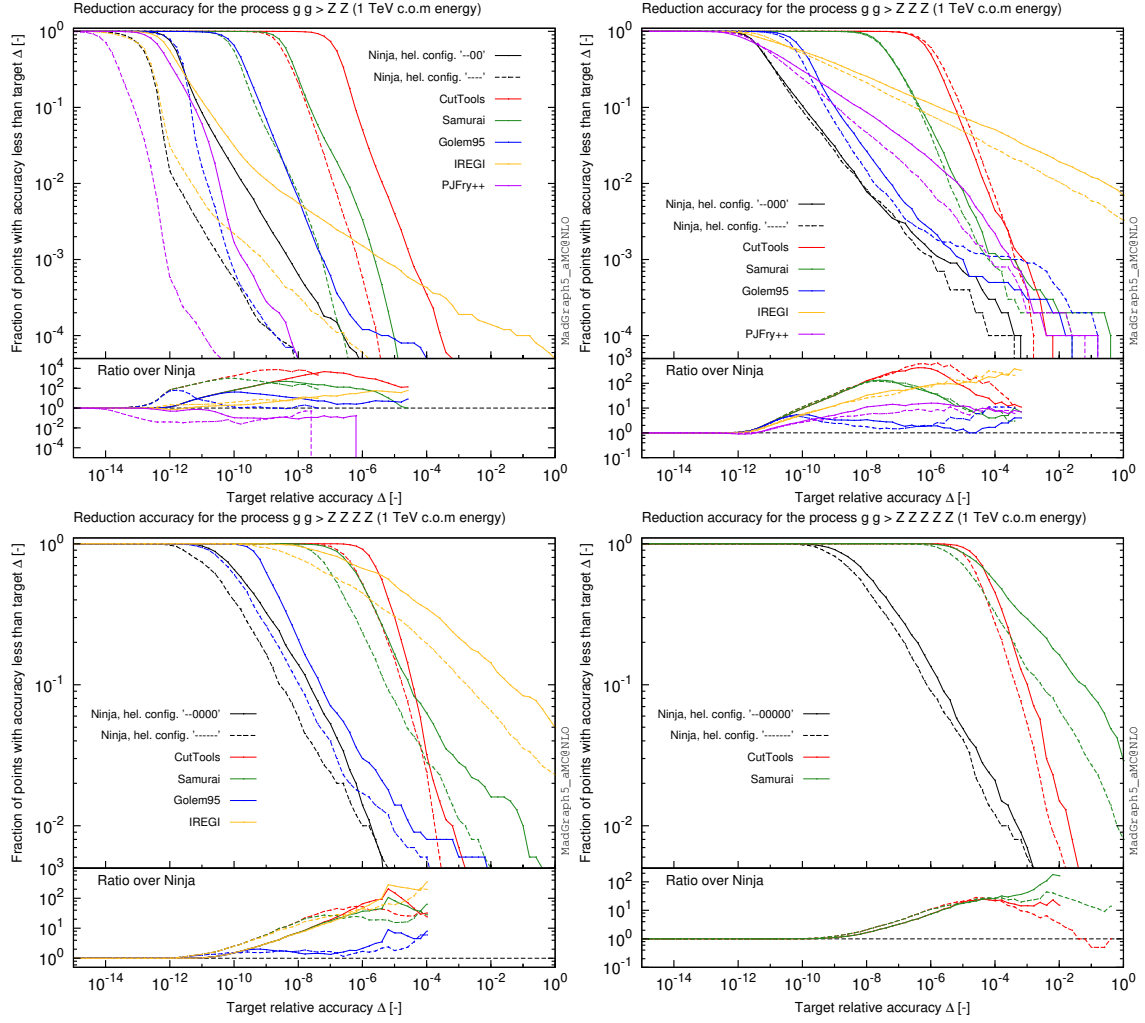


Figure 7. Same setup as in figure 2 except that no constraint on the final state transverse momenta was applied is applied in this case. The processes considered are loop-induced gluon fusion with up to five Z -bosons in the final states. Results are shown for the stability obtained when considering only a single helicity configuration where the two initial state gluons have negative helicity and the final state Z -bosons have either helicity 0 (solid line) or a negative one (dashed line).

(in terms of their tensor numerator structure and typical rank, as well as the number and spread of the different scales) can amount to variations as large as the gap between two consecutive such stability *classes*.

D Numerical results for benchmark processes

In this appendix, we include numerical evaluations of the one-loop matrix elements of the processes $gg \rightarrow t\bar{t}gg$ and $gg \rightarrow Y + \{2, 3\} \cdot g$, for selected random kinematic configurations and summed over helicity and colour assignments. Although the generation of these processes is completely automated within MG5AMC, their complexity makes them both memory and CPU intensive, so that we report here numerical results for convenience, and

also because it is, to the best of our knowledge, not available in the literature. It can serve as a benchmark for validating less automated optimized implementations, potentially fast enough to render phase-space integration feasible. We express the numerical value of the one-loop matrix element in terms of the three coefficients c_{-2} , c_{-1} and c_0 , implicitly defined as follows

$$\frac{1}{S_F} \overline{\sum_{\text{colour spin}}} 2\Re \left\{ \mathcal{A}^{(1)} \mathcal{A}^{(0)\star} \right\} = \frac{(4\pi)^\epsilon}{\Gamma(1-\epsilon)} \left(\frac{c_{-2}}{\epsilon^2} + \frac{c_{-1}}{\epsilon} + c_0 \right), \quad (\text{D.1})$$

and similarly for the Born contribution

$$\frac{1}{S_F} \overline{\sum_{\text{colour spin}}} \left| \mathcal{A}^{(0)} \right|^2 = a_0. \quad (\text{D.2})$$

The $\overline{\sum}$ symbol denotes that we sum over all final-state helicity and color configurations but we average over initial-state ones. We account for the conventional symmetry factor S_F equal to the product of the factorial of the number of identical copies of any particle in the final states (so $S_F = n_{\text{final.gluons}}!$ for the processes of interest here).

We remind the reader that numerical results like the ones presented in this appendix, along with a library for the corresponding computation for any phase-space point,⁶ can be obtained by running the following three commands in the interactive interface of MG5aMC:

```
MG5_aMC> import <chosen_model>
MG5_aMC> generate <process_definition> [virt=QCD]
MG5_aMC> launch
```

The first step is optional when considering corrections for processes within the Standard Model. For mixed Electro-Weak (EW) and QCD correction, the syntax ‘[virt=QCD QED]’ must be used instead.

D.1 Process $gg \rightarrow t\bar{t}ggg$

We specify below the chosen values for the relevant Standard Model parameters. Notice that all dimensionful quantities in this appendix D are indicated with GeV units, unless specified otherwise.

Parameter	value	Parameter	value
α_s	0.118	n_{lf}	4
m_t	173.0	m_b	4.7
Γ_t	0.0	μ_r	91.188

⁶See cp3.irmp.ucl.ac.be/projects/madgraph/wiki/MadLoopStandaloneLibrary for more information.

The kinematic configuration considered is:

E	p_x	p_y	p_z
$p_g = (500$	$, 0$	$, 0$	$, 500$
$p_g = (500$	$, 0$	$, 0$	$, -500$
$p_t = (187.1526174824760$	$, -12.57780476763981$	$, 31.11387284817540$	$, -63.01450606135616$
$p_{\bar{t}} = (289.0726872654491$	$, -68.72525125230017$	$, -217.6061267288446$	$, 39.47698029543776$
$p_g = (105.4303841594690$	$, -74.77954838939333$	$, -68.76840480238191$	$, 28.18672644397240$
$p_g = (342.3857129250602$	$, 191.9284080684538$	$, 275.1076866100379$	$, -68.60920754231151$
$p_g = (75.95859816754576$	$, -35.84580365912052$	$, -19.84702792698694$	$, 63.96000686425754$

We report below all stable digits (no rounding applied) obtained with *double precision arithmetics* for the coefficients a_0 , c_{-2} , c_{-1} and c_0 .

[GeV ⁻⁶]	$gg \rightarrow t\bar{t}ggg$
a_0	1.5516406229353492e-9
c_0	-5.6512364e-10
c_{-1}	3.504298678e-10
c_{-2}	-4.371037568250e-10

D.2 Processes $gg \rightarrow Y + \{2, 3\} \cdot g$

The details of the model that we considered including the spin-2 particle Y can be found in ref. [40]. As already noted in [44], the operator renormalization constant for the energy momentum operator is identical to unity to all orders in perturbation theory. As a result, when the graviton minimally couples to the energy momentum tensor (i.e. with $\kappa_q = \kappa_g$), there is no need for additional UV renormalisation counterterms. The parameters of this model that are relevant for the processes $gg \rightarrow Y + \{2, 3\} \cdot g$ are chosen as follows:

Parameter	value	Parameter	value
α_S	0.118	n_{lf}	5
m_t	173.0	μ_r	91.188
Γ_t	0.0	m_Y	300
$\kappa_g = \kappa_q$	1.0	Λ	1000

The $2 \rightarrow 3$ and $2 \rightarrow 4$ kinematic configurations considered are

E	p_x	p_y	p_z
$p_g = (500$	$, 0$	$, 0$	$, 500$
$p_g = (500$	$, 0$	$, 0$	$, -500$
$p_Y = (511.2453080350917$	$, 152.9697297441591$	$, 342.7230121584469$	$, -174.6796030077232$
$p_g = (328.6521936921786$	$, -16.54683898542085$	$, -313.8815119978393$	$, 96.00449450011196$
$p_g = (160.1024982727296$	$, -136.4228907587383$	$, -28.84150016060760$	$, 78.67510850761127$

and

$$\begin{aligned}
& \begin{array}{cccc} E & p_x & p_y & p_z \\ p_g = (& 500 & , & 0 & , & 0 & , & 500 &) \\ p_g = (& 500 & , & 0 & , & 0 & , & -500 &) \\ p_V = (& 187.1526174824760 & , & -12.57780476763981 & , & 31.11387284817540 & , & -63.01450606135616 &) \\ p_g = (& 105.4303841594690 & , & -74.77954838939333 & , & -68.76840480238191 & , & 28.18672644397240 &) \\ p_g = (& 342.3857129250602 & , & 191.9284080684538 & , & 275.1076866100379 & , & -68.60920754231151 &) \\ p_g = (& 75.95859816754576 & , & -35.84580365912052 & , & -19.84702792698694 & , & 63.96000686425754 &) \end{array}
\end{aligned}$$

We conclude by reporting below all stable digits obtained with *quadruple precision arithmetics* for the coefficients a_0 , c_{-2} , c_{-1} and c_0

[GeV ^{-2,-4}]	$gg \rightarrow Ygg$	$gg \rightarrow Yggg$
a_0	1.864507929648605e-3	1.529529540966328e-6
c_0	-1.003108324246923e-3	-4.0992246274118e-7
c_{-1}	8.424377254497548e-4	5.5831926923384e-7
c_{-2}	-4.201918452676584e-4	4.308749710782677e-7

Open Access. This article is distributed under the terms of the Creative Commons Attribution License ([CC-BY 4.0](https://creativecommons.org/licenses/by/4.0/)), which permits any use, distribution and reproduction in any medium, provided the original author(s) and source are credited.

References

- [1] G. Ossola, C.G. Papadopoulos and R. Pittau, *Reducing full one-loop amplitudes to scalar integrals at the integrand level*, *Nucl. Phys. B* **763** (2007) 147 [[hep-ph/0609007](#)] [[INSPIRE](#)].
- [2] R.K. Ellis, W.T. Giele and Z. Kunszt, *A Numerical Unitarity Formalism for Evaluating One-Loop Amplitudes*, *JHEP* **03** (2008) 003 [[arXiv:0708.2398](#)] [[INSPIRE](#)].
- [3] W.T. Giele, Z. Kunszt and K. Melnikov, *Full one-loop amplitudes from tree amplitudes*, *JHEP* **04** (2008) 049 [[arXiv:0801.2237](#)] [[INSPIRE](#)].
- [4] A. van Hameren, *OneLOop: For the evaluation of one-loop scalar functions*, *Comput. Phys. Commun.* **182** (2011) 2427 [[arXiv:1007.4716](#)] [[INSPIRE](#)].
- [5] T. Binoth, J.P. Guillet, G. Heinrich, E. Pilon and T. Reiter, *Golem95: A numerical program to calculate one-loop tensor integrals with up to six external legs*, *Comput. Phys. Commun.* **180** (2009) 2317 [[arXiv:0810.0992](#)] [[INSPIRE](#)].
- [6] J.P. Guillet, G. Heinrich and J.F. von Soden-Fraunhofen, *Tools for NLO automation: extension of the golem95C integral library*, *Comput. Phys. Commun.* **185** (2014) 1828 [[arXiv:1312.3887](#)] [[INSPIRE](#)].
- [7] T. Hahn and M. Pérez-Victoria, *Automatized one loop calculations in four-dimensions and D-dimensions*, *Comput. Phys. Commun.* **118** (1999) 153 [[hep-ph/9807565](#)] [[INSPIRE](#)].
- [8] R.K. Ellis and G. Zanderighi, *Scalar one-loop integrals for QCD*, *JHEP* **02** (2008) 002 [[arXiv:0712.1851](#)] [[INSPIRE](#)].

- [9] G.J. van Oldenborgh, *FF: A package to evaluate one loop Feynman diagrams*, *Comput. Phys. Commun.* **66** (1991) 1 [[INSPIRE](#)].
- [10] G. Ossola, C.G. Papadopoulos and R. Pittau, *CutTools: A program implementing the OPP reduction method to compute one-loop amplitudes*, *JHEP* **03** (2008) 042 [[arXiv:0711.3596](#)] [[INSPIRE](#)].
- [11] P. Mastrolia, G. Ossola, T. Reiter and F. Tramontano, *Scattering Amplitudes from Unitarity-based Reduction Algorithm at the Integrand-level*, *JHEP* **08** (2010) 080 [[arXiv:1006.0710](#)] [[INSPIRE](#)].
- [12] A. van Hameren, C.G. Papadopoulos and R. Pittau, *Automated one-loop calculations: A proof of concept*, *JHEP* **09** (2009) 106 [[arXiv:0903.4665](#)] [[INSPIRE](#)].
- [13] G. Bevilacqua et al., *HELAC-NLO*, *Comput. Phys. Commun.* **184** (2013) 986 [[arXiv:1110.1499](#)] [[INSPIRE](#)].
- [14] C.F. Berger et al., *An Automated Implementation of On-Shell Methods for One-Loop Amplitudes*, *Phys. Rev. D* **78** (2008) 036003 [[arXiv:0803.4180](#)] [[INSPIRE](#)].
- [15] G. Cullen et al., *Automated One-Loop Calculations with GoSam*, *Eur. Phys. J. C* **72** (2012) 1889 [[arXiv:1111.2034](#)] [[INSPIRE](#)].
- [16] F. Cascioli, P. Maierhofer and S. Pozzorini, *Scattering Amplitudes with Open Loops*, *Phys. Rev. Lett.* **108** (2012) 111601 [[arXiv:1111.5206](#)] [[INSPIRE](#)].
- [17] S. Badger, B. Biedermann and P. Uwer, *NGluon: A Package to Calculate One-loop Multi-gluon Amplitudes*, *Comput. Phys. Commun.* **182** (2011) 1674 [[arXiv:1011.2900](#)] [[INSPIRE](#)].
- [18] S. Badger, B. Biedermann, P. Uwer and V. Yundin, *Numerical evaluation of virtual corrections to multi-jet production in massless QCD*, *Comput. Phys. Commun.* **184** (2013) 1981 [[arXiv:1209.0100](#)] [[INSPIRE](#)].
- [19] G. Heinrich, G. Ossola, T. Reiter and F. Tramontano, *Tensorial Reconstruction at the Integrand Level*, *JHEP* **10** (2010) 105 [[arXiv:1008.2441](#)] [[INSPIRE](#)].
- [20] V. Hirschi, R. Frederix, S. Frixione, M.V. Garzelli, F. Maltoni and R. Pittau, *Automation of one-loop QCD corrections*, *JHEP* **05** (2011) 044 [[arXiv:1103.0621](#)] [[INSPIRE](#)].
- [21] J. Alwall et al., *The automated computation of tree-level and next-to-leading order differential cross sections and their matching to parton shower simulations*, *JHEP* **07** (2014) 079 [[arXiv:1405.0301](#)] [[INSPIRE](#)].
- [22] G. Passarino and M.J.G. Veltman, *One Loop Corrections for e^+e^- Annihilation Into $\mu^+\mu^-$ in the Weinberg Model*, *Nucl. Phys. B* **160** (1979) 151 [[INSPIRE](#)].
- [23] A.I. Davydychev, *A simple formula for reducing Feynman diagrams to scalar integrals*, *Phys. Lett. B* **263** (1991) 107 [[INSPIRE](#)].
- [24] G. Cullen et al., *Recent Progress in the Golem Project*, *Nucl. Phys. Proc. Suppl.* **205-206** (2010) 67 [[arXiv:1007.3580](#)] [[INSPIRE](#)].
- [25] J. Fleischer, T. Riemann and V. Yundin, *One-Loop Tensor Feynman Integral Reduction with Signed Minors*, *J. Phys. Conf. Ser.* **368** (2012) 012057 [[arXiv:1112.0500](#)] [[INSPIRE](#)].
- [26] P. de Aquino, W. Link, F. Maltoni, O. Mattelaer and T. Stelzer, *ALOHA: Automatic Libraries Of Helicity Amplitudes for Feynman Diagram Computations*, *Comput. Phys. Commun.* **183** (2012) 2254 [[arXiv:1108.2041](#)] [[INSPIRE](#)].

- [27] P. Mastrolia, E. Mirabella and T. Peraro, *Integrand reduction of one-loop scattering amplitudes through Laurent series expansion*, *JHEP* **06** (2012) 095 [Erratum *ibid.* **1211** (2012) 128] [[arXiv:1203.0291](#)] [[INSPIRE](#)].
- [28] D. Forde, *Direct extraction of one-loop integral coefficients*, *Phys. Rev. D* **75** (2007) 125019 [[arXiv:0704.1835](#)] [[INSPIRE](#)].
- [29] S.D. Badger, *Direct Extraction Of One Loop Rational Terms*, *JHEP* **01** (2009) 049 [[arXiv:0806.4600](#)] [[INSPIRE](#)].
- [30] T. Peraro, *Ninja: Automated Integrand Reduction via Laurent Expansion for One-Loop Amplitudes*, *Comput. Phys. Commun.* **185** (2014) 2771 [[arXiv:1403.1229](#)] [[INSPIRE](#)].
- [31] G. Cullen et al., *GoSAM-2.0: a tool for automated one-loop calculations within the Standard Model and beyond*, *Eur. Phys. J. C* **74** (2014) 3001 [[arXiv:1404.7096](#)] [[INSPIRE](#)].
- [32] H. van Deurzen, G. Luisoni, P. Mastrolia, E. Mirabella, G. Ossola and T. Peraro, *Multi-leg One-loop Massive Amplitudes from Integrand Reduction via Laurent Expansion*, *JHEP* **03** (2014) 115 [[arXiv:1312.6678](#)] [[INSPIRE](#)].
- [33] A. Denner and S. Dittmaier, *The complex-mass scheme for perturbative calculations with unstable particles*, *Nucl. Phys. Proc. Suppl.* **160** (2006) 22 [[hep-ph/0605312](#)] [[INSPIRE](#)].
- [34] A. Denner, S. Dittmaier, M. Roth and L.H. Wieders, *Electroweak corrections to charged-current $e^+e^- \rightarrow 4$ fermion processes: Technical details and further results*, *Nucl. Phys. B* **724** (2005) 247 [Erratum *ibid.* **B 854** (2012) 504] [[hep-ph/0505042](#)] [[INSPIRE](#)].
- [35] P. Draggiotis, M.V. Garzelli, C.G. Papadopoulos and R. Pittau, *Feynman Rules for the Rational Part of the QCD 1-loop amplitudes*, *JHEP* **04** (2009) 072 [[arXiv:0903.0356](#)] [[INSPIRE](#)].
- [36] P. Comon, G. Golub, L.-H. Lim and B. Mourrain, *Symmetric tensors and symmetric tensor rank*, *SIAM J. Matrix Anal. Appl.* **30** (2008) 1254 [[arXiv:0802.1681](#)].
- [37] P. Nason, S. Dawson and R.K. Ellis, *The Total Cross-Section for the Production of Heavy Quarks in Hadronic Collisions*, *Nucl. Phys. B* **303** (1988) 607 [[INSPIRE](#)].
- [38] S. Dittmaier, P. Uwer and S. Weinzierl, *NLO QCD corrections to $t\bar{t}$ + jet production at hadron colliders*, *Phys. Rev. Lett.* **98** (2007) 262002 [[hep-ph/0703120](#)] [[INSPIRE](#)].
- [39] G. Bevilacqua, M. Czakon, C.G. Papadopoulos and M. Worek, *Dominant QCD Backgrounds in Higgs Boson Analyses at the LHC: A Study of $pp \rightarrow t\bar{t} + 2$ jets at Next-To-Leading Order*, *Phys. Rev. Lett.* **104** (2010) 162002 [[arXiv:1002.4009](#)] [[INSPIRE](#)].
- [40] P. Artoisenet et al., *A framework for Higgs characterisation*, *JHEP* **11** (2013) 043 [[arXiv:1306.6464](#)] [[INSPIRE](#)].
- [41] D. de Florian, M. Grazzini and Z. Kunszt, *Higgs production with large transverse momentum in hadronic collisions at next-to-leading order*, *Phys. Rev. Lett.* **82** (1999) 5209 [[hep-ph/9902483](#)] [[INSPIRE](#)].
- [42] G. Cullen et al., *Next-to-Leading-Order QCD Corrections to Higgs Boson Production Plus Three Jets in Gluon Fusion*, *Phys. Rev. Lett.* **111** (2013) 131801 [[arXiv:1307.4737](#)] [[INSPIRE](#)].
- [43] H. van Deurzen et al., *NLO QCD corrections to the production of Higgs plus two jets at the LHC*, *Phys. Lett. B* **721** (2013) 74 [[arXiv:1301.0493](#)] [[INSPIRE](#)].

- [44] P. Mathews, V. Ravindran, K. Sridhar and W.L. van Neerven, *Next-to-leading order QCD corrections to the Drell-Yan cross section in models of TeV-scale gravity*, *Nucl. Phys. B* **713** (2005) 333 [[hep-ph/0411018](#)] [[INSPIRE](#)].
- [45] S. Karg, M. Krämer, Q. Li and D. Zeppenfeld, *NLO QCD corrections to graviton production at hadron colliders*, *Phys. Rev. D* **81** (2010) 094036 [[arXiv:0911.5095](#)] [[INSPIRE](#)].
- [46] G. Das, C. Degrande, V. Hirschi, F. Maltoni and H. Shao, *Dark matter production at the LHC with s-channel spin-2 mediator*, in preparation.
- [47] V. Hirschi and O. Mattelaer, *Automated event generation for loop-induced processes*, *JHEP* **10** (2015) 146 [[arXiv:1507.00020](#)] [[INSPIRE](#)].
- [48] E.W.N. Glover and J.J. van der Bij, *Z-boson pair production via gluon fusion*, *Nucl. Phys. B* **321** (1989) 561 [[INSPIRE](#)].
- [49] P. Agrawal and A. Shivaji, *Production of γZg and associated processes via gluon fusion at hadron colliders*, *JHEP* **01** (2013) 071 [[arXiv:1208.2593](#)] [[INSPIRE](#)].
- [50] S. Actis, A. Denner, L. Hofer, A. Scharf and S. Uccirati, *Recursive generation of one-loop amplitudes in the Standard Model*, *JHEP* **04** (2013) 037 [[arXiv:1211.6316](#)] [[INSPIRE](#)].
- [51] A. Denner and S. Dittmaier, *Reduction schemes for one-loop tensor integrals*, *Nucl. Phys. B* **734** (2006) 62 [[hep-ph/0509141](#)] [[INSPIRE](#)].
- [52] A. Denner and S. Dittmaier, *Scalar one-loop 4-point integrals*, *Nucl. Phys. B* **844** (2011) 199 [[arXiv:1005.2076](#)] [[INSPIRE](#)].
- [53] R.D. Ball et al., *Parton distributions with LHC data*, *Nucl. Phys. B* **867** (2013) 244 [[arXiv:1207.1303](#)] [[INSPIRE](#)].